

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

Charles P. Schaum[†]

Released 2016/10/26

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

1	Introduction	2	2.7.4	Index Cross-References	26
1.1	Preliminary Information . . .	2	2.7.5	Index Sorting	27
1.2	Technical Notes	3	2.7.6	Index Tags	29
1.3	What’s In A Name?	4	2.8	“Text Tags”	30
2	Usage	5	2.9	Name Decisions	31
2.1	Quick Start Guide	5	2.9.1	Testing Decisions . . .	31
2.1.1	Main Interface	5	2.9.2	Changing Decisions . .	34
2.1.2	Simplefied Interface . . .	8	2.10	Name Variant Macros	35
2.1.3	Older Syntax	10	2.11	Longer Examples	41
2.2	Package Options	11	2.11.1	Variant Spellings . . .	41
2.3	Naming Macros	14	2.11.2	\LocalNames	42
2.3.1	\Name and \Name* . . .	14	2.11.3	Unicode and NFSS . . .	43
2.3.2	Forenames: \FName . .	15	2.11.4	L ^A T _E X Engines	45
2.4	Language Issues	16	2.11.5	Hooks: Intro	46
2.4.1	Affixes Need Commas	16	2.11.6	Hooks: Life Dates . .	47
2.4.2	Eastern Names	17	2.11.7	Hooks: Advanced . . .	49
2.4.3	Initials	18	2.11.8	Full Redesign	53
2.4.4	Hyphenation	18	2.12	Naming Pattern Reference . .	54
2.4.5	Listing by Surname . .	19	2.13	Errors and Warnings	58
2.4.6	Particles	19	3	Implementation	59
2.4.7	Accented Names . . .	21	3.1	Flags and Registers	59
2.4.8	Non-English Format . .	21	3.2	Hooks	61
2.5	Spaces and Punctuation . . .	22	3.3	Package Options	62
2.6	Formatting in the Text	23	3.4	Internal Macros	62
2.7	Indexing Macros	25	3.5	User Interface Macros	71
2.7.1	Indexing Control . . .	25	4	Change History	92
2.7.2	Indexing and babel . .	25	5	Index	94
2.7.3	Index Entries	25			

*This file describes version v3.0, last revised 2016/10/26.

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

1 Introduction

1.1 Preliminary Information

Disclaimer

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

Denotative Signs

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

Design

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

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

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

Thanks

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

1.2 Technical Notes

About the package itself:

- Most current changes are compatible with older versions.
- The package works with `xindy` and `makeindex`. We recommend `xindy` for languages whose collating sequences do not map to English.¹
- 3.0** • Notable changes, features, and fixes that correspond to version numbers are indicated in the margin.
- 3.0** • We support Eastern alternate names when using “native” format. We simplify indexing, custom formatting, and other tasks.
- 3.0** • Name output, index page entries, and index cross-references are independent due to modular design.
- 3.0** • Warnings for the indexing macros are suppressed unless one uses the `verbose` option. The `nameauth` environment will produce warnings.
- 2.6** • The `comma` option and the old syntax are no longer restrictive, save with `\AKA` and its derivatives. See Sections 2.1.3, 2.4.1, and 2.10.
- 2.5** • No formatting is selected by default. To implement formatting, see Sections 2.4.8, 2.6, and 2.11.5f.

About the manual:



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

About package building:

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

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

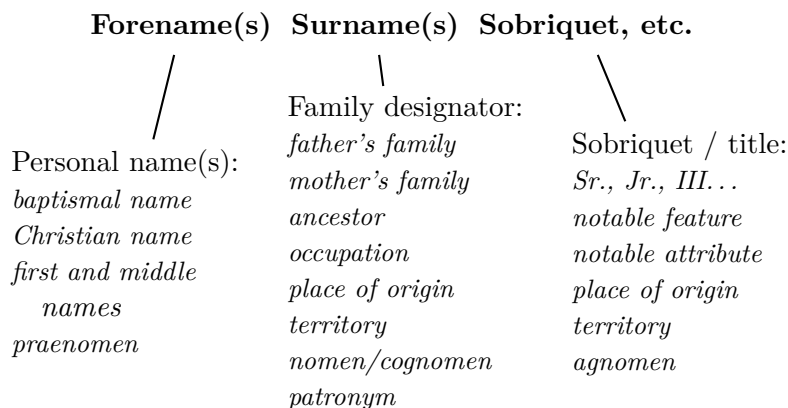
²The manual is used as the test suite. In dvi mode the manual omits all references to *TikZ* because some dvi display programs (*e.g.* `dviout`, but not `xdvi`) will emit errors about bad specials even if one just includes the `tikz` package. The *TikZ* diagrams herein will appear as blank space in that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

1.3 What's In A Name?

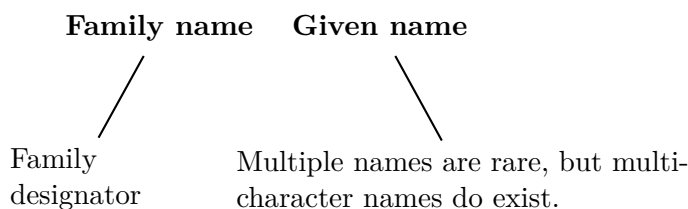
Name forms are ambiguous apart from historical and cultural contexts. In this manual we refer to three general classes of names, shown below. Section 2.1 shows how these classes are implemented.

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

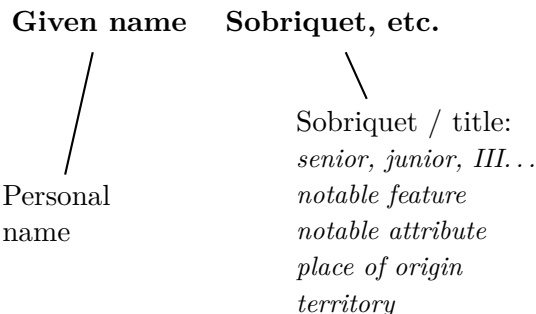
1. Western name:



2. Eastern name:



3. Ancient name:



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

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

2 Usage

2.1 Quick Start Guide

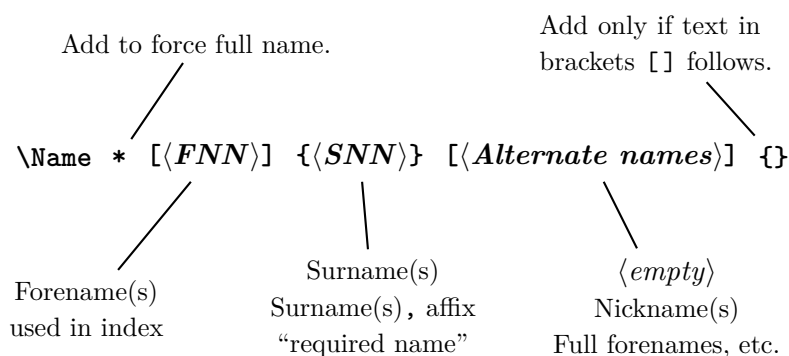
Here we offer templates for the most common macros and features. The details will come later. This section also formats the first uses of names in **boldface** to illustrate how first uses are long, but long uses need not be first.

2.1.1 Main Interface

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

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

Western Names



Examples:

One always must include all fields for consistent index entries.

```

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

```

When using $\langle Alternate names \rangle$, the $\langle FNN \rangle$ argument must have a name in it. The $\langle Alternate names \rangle$ are swapped with the $\langle FNN \rangle$, but only in the body text:

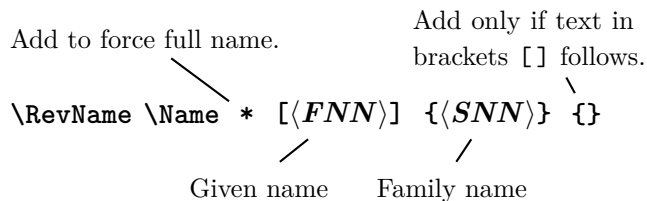
```

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

```

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

Eastern Names in the Text, Western Index Entry

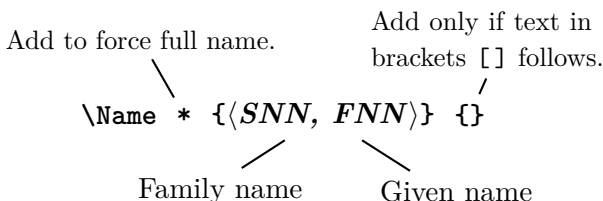


These are Western name forms without affixes that the reversing macros (Section 2.4.2) cause to display in Eastern order in the body text only. We indicate them with a dagger.

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

The index entries are Western in fashion: $\langle SNN \rangle$, $\langle FNN \rangle$. This “non-native” form of Eastern names conflicts with both comma-delimited forms and the old syntax.

Eastern Names in the Text, Eastern Index Entry



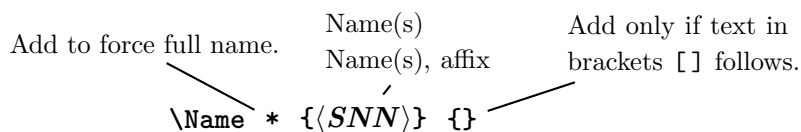
Examples:

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

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

These names truly are Eastern names. They always take the form $\langle SNN FNN \rangle$ in the index. See Section 2.4.2. We call this the “native” Eastern form. For the old syntax, see Section 2.1.3.

Ancient Names



Examples:

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

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

Macro Patterns:

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

$\langle prefix macros \rangle$	<code>\Name</code>	$\langle arguments \rangle$
$\langle prefix macros \rangle$	<code>\Name*</code>	$\langle arguments \rangle$
$\langle prefix macros \rangle$	<code>\FName</code>	$\langle arguments \rangle$
	<code>\IndexName</code>	$\langle arguments \rangle$
	<code>\IndexRef</code>	$\langle arguments \rangle \langle target \rangle$
	<code>\ExcludeName</code>	$\langle arguments \rangle$
	<code>\IncludeName</code>	$\langle arguments \rangle$
	<code>\IncludeName*</code>	$\langle arguments \rangle$
	<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:

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

Most $\langle prefix macros \rangle$ are “stackable” in front of the naming macros; for example, `\CapName\RevName\Name...`

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

2.1.2 Simplified Interface

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

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

```
\begin{nameauth}
  \< \langle cseq1 \rangle & \langle FNN \rangle & \langle SNN \rangle & > % West./Non-native East.
  \< \langle cseq2 \rangle & \langle FNN \rangle & \langle SNN, affix \rangle & > % Western
  \< \langle cseq3 \rangle & & \langle SNN \rangle & > % ancient/mono
  \< \langle cseq4 \rangle & & \langle SNN, affix \rangle & > % royal/ancient
  \< \langle cseq5 \rangle & & \langle SNN, FNN \rangle & > % Native Eastern
  \< \langle cseq6 \rangle & & \langle SNN \rangle & \langle FNN or affix \rangle > % old syntax6
\end{nameauth}
```

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

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

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

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

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

⁶See Section 2.1.3.

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

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

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

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

WESTERN:	ANCIENT / MONONYM
\Wash George Washington	\Aris Aristotle
\LWash George Washington	\Aris Aristotle
\Wash Washington	
\SWash George	MEDIEVAL/ROYAL:
\RevComma\LWash	\Eliz Elizabeth I
. Washington, George	\Eliz Elizabeth
	\Attil Attila the Hun
	\Attil Attila
PARTICLES: (Section 2.4.6)	
\Soto Hernando de Soto	“NON-NATIVE” EASTERN:
\Soto de Soto	\Konoe Fumimaro Konoe†
\CapThis\Soto De Soto	\LKonoe . Fumimaro Konoe†
	\Konoe Konoe†
AFFIXES: (Section 2.4.1)	\SKonoe Fumimaro†
\JR II . John David Rockefeller II	\CapName\RevName\LKonoe
\LJR II John David Rockefeller II KONOE Fumimaro†
\DropAffix\LJR II	\CapName\Konoe . KONOE†
. John David Rockefeller	
\JR II Rockefeller	“NATIVE” EASTERN:
\SJR II John David	\CapName\Yamt
	. YAMAMOTO Isoroku
NICKNAMES: (Section 2.3.2)	\CapName\LYamt
\JR IV John David Rockefeller IV YAMAMOTO Isoroku
\LJR IV [Jay] Jay Rockefeller IV	\CapName\Yamt
\SJR IV [Jay] Jay YAMAMOTO
\DropAffix\LYayR . . Jay Rockefeller	\RevName\LYamt
\SJayR Jay Isoroku Yamamoto
\Lewis Full . . Clive Staples Lewis	\SYamt Yamamoto
\LLewis C.S. Lewis	\ForceFN\SYamt
\Lewis Lewis Isoroku
\SLewis C.S.	



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

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

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

2.1.3 Older Syntax

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

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

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

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

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



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

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



2.2 Package Options

One includes the `nameauth` package thus:

```
\usepackage[<option1>,<option2>,...] {nameauth}
```

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

Default options are shown in boldface.

Choosing Features

Enable Package Warnings

verbose Show warnings about index cross-references.

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

Choose Formatting

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

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

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

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

Enable/Disable Indexing

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

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

Enable/Disable Index Sorting

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

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

Affect the Syntax of Names

Show/Hide Affix Commas

<code>nocomma</code>	Suppress commas between surnames and affixes, following the <i>Chicago Manual of Style</i> and other conventions.
<code>comma</code>	Retain commas between surnames and affixes.

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

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

Capitalize Entire Surnames

<code>normalcaps</code>	Do not perform any special capitalization.
<code>allcaps</code>	Capitalize entire surnames, such as romanized Eastern names.

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



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

Reverse Name Order

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

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

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

Typographic Post-Processing



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

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

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

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

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

Formatting Attributes

- | | |
|------------------------|--|
| <code>smallcaps</code> | Set the first use of an entire name in small caps. ¹¹ |
| <code>italic</code> | Set the first use of an entire name in italic. |
| <code>boldface</code> | Set the first use of an entire name in boldface. |
| <code>noformat</code> | Do not define a default format. |

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

2.5 Current versions assign no default formatting to names.

⁸This 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.

⁹I 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.

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

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

2.3 Naming Macros

2.3.1 `\Name` and `\Name*`

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

```
\Name [ $\langle FNN \rangle$ ]{ $\langle SNN$ , opt.  $\langle FNN/Affix \rangle$ }[ $\langle Alternate names \rangle$ ]
\Name* [ $\langle FNN \rangle$ ]{ $\langle SNN$ , opt.  $\langle FNN/Affix \rangle$ }[ $\langle Alternate names \rangle$ ]
```

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

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

<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 \Miyazaki</code>	Miyazaki Hayao
<code>\Name*[Miyazaki, Hayao][Sensei]</code>	Miyazaki Sensei
<code>\Name {Miyazaki, Hayao} or \Miyazaki</code>	Miyazaki
<code>\Name {Charles, the Bald} or \CBald</code>	Charles the Bald
<code>\Name*[Charles, the Bald] or \LCBald</code>	Charles the Bald
<code>\Name {Charles, the Bald} or \CBald</code>	Charles

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

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

2.3.2 Forenames: \FName

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

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

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

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

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

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

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



Using “default nicknames” in the simplified interface has some caveats. The first use `\Ches` prints “Chesley B. Sullenberger III.” Later, `\SChes` and `\SSully` print “Chesley B.” and “Sully.” While `\SChes[Sully]` always gives “Sully,” `\SSully[Chesley B.]` prints “Sully[Chesley B.]” With `\Sully`, the *⟨Alternate names⟩* field is already occupied, making `[Chesley B.]` normal text.

2.4 Language Issues

2.4.1 Affixes Need Commas

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

<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



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

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

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

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

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

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

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

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

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

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

2.4.2 Eastern Names

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

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

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

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

```
\Name{\Eastern SNN, Eastern FNN}
\Name{\Eastern SNN, Eastern FNN}{\Alternate names}
\Name{\Eastern SNN}{\Eastern FNN} (old syntax)
```

The index entry of this name form looks like $\langle SNN \rangle$ $\langle FNN \rangle$ (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 2.2), 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[Minister]</code>	Minister Konoe†	<i>\not appropriate</i>
<code>\Konoe</code>	Konoe†	Konoe†
<code>\SKonoe</code>	Fumimaro†	Fumimaro†
<code>\LYamt</code>	Yamamoto Isoroku	Isoroku Yamamoto
<code>\LYamt[Admiral]</code>	<i>\not appropriate</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 2.4.5) only makes sense with Western names and maybe non-native Eastern names, but not with native Eastern names or ancient forms. That is why native Eastern forms and ancient forms are unaffected by the commatized form of reversing.

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

`\AllCapsActive` 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*{\langle SNN, Affix \rangle}`.

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

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

	<i>unchanged</i>	<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 2.4.8 and 2.11.7.

2.4.3 Initials

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

<code>\PretagName[E.\,B.]{White}% {White, E. B.}</code>	<code>\White</code> and <code>\LWhite</code>	E. B. White
<code>\begin{nameauth} < White & E.\,B. & White & > \end{nameauth}</code>	Normal text:	E. B. White

2.4.4 Hyphenation

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

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

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

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

2.4.5 Listing by Surname

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

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

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

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

<code>\LKonoe</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`.

2.4.6 Particles

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

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

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

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

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

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



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

<code>\Name[Catherine]{\textsc{de'~Medici}}</code>	Catherine DE' MEDICI
<code>\textsc{De'~Medici}%</code>	
<code>\IndexName[Catherine]{\textsc{de'~Medici}}</code>	DE' MEDICI

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

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

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

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

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

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

Non-English contexts do not necessarily bind particles to surnames. Using `\Name` and `\FName` with alternate forenames helps address this and may skirt the particle capitalization issue. See also Section 2.12.

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

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

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

2.4.7 Accented Names

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



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

<code>\Name[Johann]{Andre\"a}</code>	Johann Andrea
<code>\Name[Johann]{Andreä}</code>	Johann Andreä instead of Andrea
<code>\Name{\AE thelred, II}</code>	Æthelred II
<code>\Name{Æthelred, II}</code>	Æthelred II instead of Æthelred

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

2.4.8 Non-English Format

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

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

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

```

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

```

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

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

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

```

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

```

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

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

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

\Fukuyama	FUKUYAMA Takeshi
\OFukuyama	FUKUYAMA
\LOFukuyama	FUKUYAMA Takeshi
\Fukuyama	FUKUYAMA

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

2.5 Spaces and Punctuation

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

<i>Macro Example</i>	<i>Resulting Text</i>
\Name*[Martin Luther]{King, Jr.}	Martin Luther King Jr.
\Name*[_ _Martin Luther]{_ _King, Jr.}	Martin Luther King Jr.
\Name*[Martin Luther_ _]{King, Jr._ _}	Martin Luther King Jr.
\Name*[_ Martin Luther_]{_ King, Jr._}	Martin Luther King Jr.
\Name*[Martin_ _ _Luther]{King_ _ _Jr.}	Martin Luther King Jr.

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

<i>Macro Example</i>	<i>periods</i>	<i>Resulting Text</i>
\Name[Martin Luther]{King, Jr.}.	2 → 1	Martin Luther King Jr.
\Name[Martin Luther]{King, Jr.}.	2 → 1	King.
\Name[Martin Luther]{King, Jr.}_	1 → 0	King
\Name*[Martin Luther]{King, Jr.}.	2 → 1	Martin Luther King Jr.
\Name*[Martin Luther]{King, Jr.}_	1 → 1	Martin Luther King Jr.

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

2.6 Formatting in the Text

There are two kinds of formatting at work:

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

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

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

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

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

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

Now we switch to the “front matter” mode:

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

Then we switch back to “main matter” mode:

```
\NamesActive
\Name[Rudolph]{Carnap}      RUDOLPH CARNAP
\Name[Rudolph]{Carnap}      Carnap
\Name[Nicolas]{Malebranche} NICOLAS MALEBRANCHE
\Name[Nicolas]{Malebranche} Malebranche
```



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

- Using `alwaysformat` in the front matter will produce: **Albert Einstein**, then **Einstein**; **Confucius**, then **Confucius**.
- Using `alwaysformat` in the main matter will produce: MARCUS TULLIUS CICERO, then CICERO; CHARLES THE BALD, then CHARLES.



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

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

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

```
\bgroup<Hook>{#1}\egroup
```



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

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

The problem above is that JOHN MAYNARD KEYNES in the text affects formatting in the footnotes.¹⁵ Using `\ForgetName` to address this can be tedious.

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

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

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

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

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

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

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

2.7 Indexing Macros

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

2.7.1 Indexing Control

`\IndexActive` Using the `noindex` option deactivates the indexing function of this package until `\IndexInactive` occurs. Another macro, `\IndexInactive`, will deactivate indexing again. These can be used throughout the document. They work differently than `\ExcludeName` and `\IncludeName`, which leverage the cross-referencing system. **`\IndexInactive` also suppresses index sorting and tagging macros.**

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

2.7.2 Indexing and `babel`

`texindy` Using `babel` with Roman page numbers will put `\textlatin` in the index entries if one includes a language that does not use the Latin alphabet—even if the main language does. The `texindy` program will ignore such references. This issue can affect `nameauth`.

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

```
\newcommand\fixindex[1]{\def\textlatin##1{##1}\#1}
...
\fixindex{ <paragraphs of running text> }
```

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

2.7.3 Index Entries

`\IndexName` The naming macros (`\Name`, etc.) use this macro to create index entries. Direct use of `\IndexName` is available also to users. It prints nothing in the body text. The syntax is:

```
\IndexName[<FNN>]{<SNN>}[<Alternate names>]
```

`\IndexName` complies with the new syntax, where a suffixed pair in `<SNN>` is a name/affix pair that can be ancient or Eastern. If `<FNN>` are present, it ignores `<Alternate names>` for Western and native Eastern name forms. Otherwise, if `<FNN>` are absent, `\IndexName` sees `<Alternate names>` as an affix or Eastern forename using the old syntax.

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

The indexing mechanism in the `nameauth` package follows *Chicago Manual of Style* standards regarding Western names and affixes. Thus the name Chesley B. Sullenberger III becomes “Sullenberger, Chesley B., III” in the index.

2.7.4 Index Cross-References

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

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

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

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

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

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

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

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

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

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

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

¹⁷When 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.

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

Kris Kringle, then Kringle.

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

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

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

3.0

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

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

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

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

2.7.5 Index Sorting

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

Basic Sorting (Makeindex)

`\PretagName` The `nameauth` package integrates this sort of index sorting automatically by
2.0 using a “pretag.” The syntax is:

```
\PretagName [FNN]{SNN}[Alternate names]{<tag>}
```

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

```
\PretagName[Jan]{Łukasiewicz}{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 created. If you create a cross-reference with `\AKA` and you want to “pretag” it, see Section 2.10.

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

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

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

Extra Spaces and Sorting



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

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

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

```
NFSS: \index{Fußball} → \indexentry{Fu\IeC_{\ss}ball}{\page}
fontspec: \index{Fußball} → \indexentry{Fußball}{\page}
csseq: \index{Fu\ss ball} → \indexentry{Fu\ss_ball}{\page}
```

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

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

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

```
\usepackage{verbatim}
\newif\ifdebug
\ifdebug
  \verbatiminput{\jobname.ind}
\fi
```

2.7.6 Index Tags

\TagName This macro creates an index tag that will be appended to all index entries for a corresponding **\Name** from when it is invoked until the end of the document or a corresponding **\UntagName**. Both **\TagName** and **\UntagName** handle their arguments like **\IndexName**. If global tags are desired, tag names in the preamble.

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

Tags are not “pretags.” To help sort that out, we look at what gets affected by these commands:

\index{	\PretagName	\TagName and \UntagName
Aethelred 20	Ethelred II	, king}

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

Tags created by **\TagName** can be helpful in the indexes of history texts, as can other package features. Below, **\TagName** causes the **nameauth** indexing macros to append “**,_Lpope**” to the index entries for Gregory I and Leo I:

```
\TagName{Leo, I}{, pope}      (in the preamble)
\TagName{Gregory, I}{, pope}
...
\Name*{Leo, I} was known as    Leo I was known as
\AKA{Leo, I}{Leo}[the Great].  Leo the Great.
...
\Name{Gregory, I} ‘\ForceFN%   Gregory “the Great,”
\AKA*{Gregory, I}{Gregory}%    another pope.
[the Great],’ another pope.
```

Tags are literal text that can be daggers, asterisks, and even specials. For example, all fictional names in the index of this manual are tagged with an asterisk. One must add any desired spacing to the start of the tag. Tagging aids scholarly indexing and can include life/regnal dates and other information.

\TagName works with all name types, not just medieval names. Back in Section 2.1 we had the example of Jimmy Carter (cross-reference in the index). **\TagName** adds “**,_Lpresident**” to his index entry.



You can use the **{⟨tag⟩}** field of **\TagName** to add “specials” to index entries for names. Every name in this manual is tagged with at least **|hyperpage** to allow hyperlinks in the index using the **ltxdoc** class and **hypdoc** package. You may have to use **\string** before **|hyperpage** where a vertical bar is active. For example, the following will produce a special index entry with no page number:

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

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

`\UntagName` `\TagName` will replace one tag with another tag, but it does not remove a tag from a name. That is the role of `\UntagName`. The syntax is:

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

By using `\TagName` and `\UntagName`, one can disambiguate different people with the same name. For example, using macros from Section 2.9.2:

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

This refers to John Smith.
Now another John Smith.
Then a third John Smith.
Then the first John Smith.
```

The tweaking macros `\ForgetName` and `\SubvertName` make it seem like you are dealing with three people who have the same name. The index tags will group together those entries with the same tag.¹⁹

2.8 “Text Tags”

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

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

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

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

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

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

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

Thus, ‘`\NameQueryInfo[George]{Washington}.`’ expands to “(1732–99).” As with index tags, one can put a space at the start of a tag—or not, depending on the use. Sections 2.7.6 and 2.11.5f. illustrate how this can permit tags like asterisks, daggers, and footnotes, such as one for Schuyler Colfax.²⁰

¹⁹Since this document, unlike the example above, puts an asterisk by all fictional names in the index, it puts an asterisk at the beginning of the tags above and does not `\UntagName` John Smith, but retags him with an asterisk again.

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

The source for the previous example looks like:

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

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

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

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

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

```
\NameClearInfo[<FNN>]{<SNN>}[<Alternate names>]
```

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

2.9 Name Decisions

2.9.1 Testing Decisions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This macro works the same as `\IfMainName`. A “front matter” name is created by the naming macros when the `frontmatter` option is used or after `\NamesInactive`. It is distinguished from those names that occur in the main matter and those that have been used as cross-references.

For example, based on Section 2.6, we see that “Carnap is both” a formatted and unformatted name with the following test:

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

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

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

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

This macro works similarly to `\IfMainName`, although it has an additional `⟨excluded⟩` branch in order to detect those names excluded from indexing by `\ExcludeName` (Section 2.7.4).

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

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

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

4. This gives us “Ventura has an alias.”

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

A similar use of `\IfAKA{Confucius}` tells us that “Confucius is not an alias.” Yet we should test that completely:

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

Here we test for a name used with `\ExcludeName` (Section 2.7.4) to get the result, “Grinch is excluded”:

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

2.9.2 Changing Decisions

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

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

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

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

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

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

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

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

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

`Scope` The default behavior of these two macros changes whether a name is “forgotten” or “subverted” simultaneously for front matter and main matter names. Remember the example on page 32 above that gave us the answer, “Carnap is both”? Now watch closely: After we use `\ForgetName[Rudolph]{Carnap}` we get the result: “Rudolph Carnap is not mentioned.” Both the main matter name and the front matter name were forgotten!

This default behavior helps synchronize formatted and unformatted types of names. For example, if you wanted to use unformatted names in the footnotes and formatted names in the text (Section 2.6), you could use, *e.g.* `\SubvertName` right after the first use of a name in the body text, ensuring that all references in the text and notes would be short unless otherwise modified.²¹

`\LocalNames` `\GlobalNames` If, however, this “global” behavior of `\ForgetName` and `\SubvertName` is not desired, you can use `\LocalNames` to change that behavior and `\GlobalNames` to restore the default behavior. Both of these macros work globally.

After `\LocalNames`, if you are in a “front matter” section (the `frontmatter` option or `\NamesInactive`) `\ForgetName` and `\SubvertName` will only affect unformatted names. If you are in a “main matter” section via the `mainmatter` option or `\NamesActive`, then `\ForgetName` and `\SubvertName` will only affect formatted names. Section 2.11.2 offers a long example.

²¹This manual takes advantage of that behavior at times in order to synchronize first and subsequent uses of names between formatted and unformatted sections of the body text.

2.10 Name Variant Macros

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

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

```
\AKA [ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ }[ $\langle Alt. FNN \rangle$ ]{ $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ]
\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`.** That is due to the following problem:

$$\begin{array}{c} [\langle FNN_1 \rangle] \{ \langle SNN_1 \rangle \} \quad \Big| \quad [\langle Alt_1 \rangle] [\langle FNN_2 \rangle] \quad \Big| \quad \{ \langle SNN_2 \rangle \} [\langle Alt_2 \rangle] \\ \text{Which to pick?} \end{array}$$

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. See also Section 2.7.6.

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

`\AKA` prints the $\langle Alt. FNN \rangle$ 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 `\PName`, meaning that `\ForceFN` works with it in the same manner. For the older behavior of `\AKA*` use the `oldAKA` option or always precede `\AKA*` with `\ForceFN`.

Basic Operation

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

<code>\AKA</code>	$[\langle FNN \rangle] \{ \langle SNN \rangle \}$...	<i>Western</i>
<code>\AKA</code>	$[\langle FNN \rangle] \{ \langle SNN, Affix \rangle \}$...	
<code>\AKA</code>	$\{ \langle SNN \rangle \}$...	<i>Ancient</i>
<code>\AKA</code>	$\{ \langle SNN, Affix \rangle \}$...	
<code>\AKA</code>	$\{ \langle SNN, FNN \rangle \}$...	<i>Eastern</i>

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

<code>\AKA</code>	...	$[\langle Alt. FNN \rangle] \{ \langle Alt. SNN \rangle \}$	<i>Western</i>
<code>\AKA</code>	...	$[\langle Alt. FNN \rangle] \{ \langle Alt. SNN \rangle \} [\langle Alt. names \rangle]$	
<code>\AKA</code>	...	$[\langle Alt. FNN \rangle] \{ \langle Alt. SNN, Affix \rangle \}$	
<code>\AKA</code>	...	$[\langle Alt. FNN \rangle] \{ \langle Alt. SNN, Affix \rangle \} [\langle Alt. names \rangle]$	
<code>\AKA</code>	...	$\{ \langle Alt. SNN \rangle \}$	<i>Ancient</i>
<code>\AKA</code>	...	$\{ \langle Alt. SNN, Affix \rangle \}$	
<code>\AKA</code>	...	$\{ \langle Alt. SNN, Alt. FNN \rangle \}$	<i>Eastern</i>

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

<code>\AKA</code>	...	$\{ \langle Alt. SNN, Alt. FNN \rangle \} [\langle Alt. names \rangle]$	<i>Eastern</i>
<code>\AKA</code>	...	$\{ \langle Alt. SNN, Alt. Affix \rangle \} [\langle Alt. names \rangle]$	<i>Ancient</i>

These alternate-name forms use the old syntax:

<code>\AKA</code>	...	$\{ \langle Alt. SNN \rangle \} [\langle Alt. FNN \rangle]$	<i>Eastern</i>
<code>\AKA</code>	...	$\{ \langle Alt. SNN \rangle \} [\langle Alt. Affix \rangle]$	<i>Ancient</i>

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

<code>\AKA[Bob]{Hope}[Leslie Townes]{Hope}</code>	Leslie Townes Hope
<code>\AKA*[Bob]{Hope}[Leslie Townes]{Hope}</code>	Leslie Townes
<code>\AKA[Bob]{Hope}%</code> <code>[Leslie Townes]{Hope}[Leslie]</code>	Leslie Hope
<code>\AKA*[Bob]{Hope}%</code> <code>[Leslie Townes]{Hope}[Leslie]</code>	Leslie

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

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

<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 Er

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

<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}%</code> <code>{Gregory}[the Great]</code>	the Great

Formatting Alternate Names: General

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

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

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

formatAKA



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

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

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

`formatAKA`

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

`\Format\AKA{Elizabeth, I}[Good Queen]{Bess}` *Good Queen Bess*.

Main Matter: ELIZABETH I was known as “Good Queen Bess.” Again we mention Queen Elizabeth, “Good Queen Bess.”

`\Format\AKA{Elizabeth, I}[Good Queen]{Bess}` GOOD QUEEN BESS.

`alwaysformat`

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

Main Matter: ELIZABETH I was known as “GOOD QUEEN BESS.” Again we mention Queen ELIZABETH, “GOOD QUEEN BESS.”

Formatting Alternate Names: Continental

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

1. Tag the names for proper sorting.

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

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

`{Ruehmann, Heinrich Wilhelm}%`

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

`\AKA*[Heinz]{\textsc{Rühmann}}%`

`[Heinrich Wilhelm]{\textsc{Rühmann}} %`

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

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

4. `\FName` prints “Heinz.”

`‘‘\FName[Heinz]{\textsc{Rühmann}}’’ %`

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

`\Name[Heinz]{\textsc{Rühmann}} (7 March 1902\,--\,3%`

`October 1994) was a German actor in over 100 films.`

The resulting text is:

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

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:²²

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

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

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

- 3.0** • For whatever name you use for the *see also* reference, put the cross-reference after all of the page references. For example, you could put both of these macros at the end of the document:²³

```
\SeeAlso\IndexRef{Maimonides}{Rambam}  
\SeeAlso\IndexRef{Rambam}{Maimonides}
```

- You could let the last reference to either name be handled by `\SeeAlso\AKA`, but that could be more confusing and prone to error.

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

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

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

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

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

²³Different standards exist for punctuating index entries and cross-references. Check with your publisher, style guide, docs for `xindy` and `makeindex`, and <http://tex.stackexchange.com>.

General Tips for \AKA

- `[\FNN]{\SNN}` is the target name. `[\Alt. FNN]{\Alt. SNN}{\Alt. names}` is the cross-reference to the target. Neither create an index entry with page references for the target.
- The old syntax causes `\AKA` and `\AKA*` to fail: `\AKA{Louis}[XIV]{Sun King}` and `\AKA{Gregory}[I]{Gregory}[the Great]`.
- The `\Alt. SNN` field uses comma-delimited suffixes.
- The `\Alt. names` field does not use comma-delimited suffixes.
- Eastern names work as pseudonyms, with all that entails. One can refer to Lafcadio Hearn as KOIZUMI Yakumo:
`\CapName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}`.
- Particles work: Du Cange is the alternate name for Charles du Fresne, which is capitalized via `\CapThis\AKA`. See also Section 2.12.
- Reversing works, *e.g.*,
`\RevComma\AKA...: Hope, Leslie Townes`
`\RevName\AKA... : Yakumo KOIZUMI`



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

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



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

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



- Special cases like “Iron Mike” Tyson as the nickname for Mike Tyson may be handled in a number of ways.
 - Follow “‘Iron Mike’” with `\IndexName[Mike]{Tyson}` and do whatever you want in the text. This may be the easiest solution.
 - Use “‘\AKA[Mike]{Tyson}{Iron Mike}’” to create “Iron Mike” in the text and a cross-reference to “Tyson, Mike” in the index. Be sure to have an occurrence of `\Name[Mike]{Tyson}` in the text. See also Section 2.10. This is the best solution in terms of how `nameauth` is designed.
 - Always get “Iron Mike Tyson” with something like:
`\newcommand*\Iron{\SubvertName[Mike]{Tyson}%`
`\FName[Mike]{Tyson}[Iron Mike] \Name[Mike]{Tyson}`
 “‘\Iron’” gives you “Iron Mike Tyson.” You are responsible for typesetting the first use and creating a cross-reference. This solution runs somewhat contrary to the design principles of `nameauth`, but it may be helpful if you want the invariant name “Iron Mike Tyson” to recur and you want to save typing.

And the Silliest Macro Comes Last

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

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

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

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

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

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

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

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

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

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



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

2.11 Longer Examples

2.11.1 Variant Spellings



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

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

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

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

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

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

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

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

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

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

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

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

2.11.2 \LocalNames

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

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

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

Next we create a formatted name in the “main matter”:

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

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

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

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

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

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

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

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

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

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

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

2.11.3 Unicode and NFSS

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

À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	SMALL CAPS
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	normal
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß	SMALL CAPS
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß	normal
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	SMALL CAPS
à á â ã ä å æ	ç è é ê ë	ì í î ï ð ñ	normal
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	SMALL CAPS
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	normal
Ǻ ǻ Ǽ Ǿ ǿ ǿ ǿ	Ǿ ǿ Ǿ Ǿ Ǿ Ǿ Ǿ	Ǿ Ǿ Ǿ Ǿ	SMALL CAPS
Ǻ ǻ Ǽ Ǿ ǿ ǿ ǿ	Ǿ ǿ Ǿ Ǿ Ǿ Ǿ Ǿ	Ǿ Ǿ Ǿ Ǿ	normal
IJ iJ L l L l	Ŋ ŋ Ŋ ŋ Ŋ ŋ	Ŕ ŕ Ŗ ŗ	SMALL CAPS
IJ ij L l L l	Ŋ ŋ Ŋ ŋ Ŋ ŋ	Ŕ ŕ Ŗ ŗ	normal
Š š Š š Ţ ť Ţ ť	Ũ ũ Ũ ũ	Ž ž Ž ž Ž ž	SMALL CAPS
Š š Š š Ţ ť Ţ ť	Ũ ũ Ũ ũ	Ž ž Ž ž Ž ž	normal



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

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



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



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

Argument	Macro	Result
abc	<code>\foo abc\relax</code>	<code><ab><c></code>
<code>{æ}bc</code>	<code>\foo {æ}bc\relax</code>	<code><æb><c></code>
<code>\aebc</code>	<code>\foo \ae bc\relax</code>	<code><æb><c></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><æb><c></code>
<code>æbc</code>	<code>\foo æbc\relax</code>	<code>lualatex</code>	<code><æb><c></code>
<code>æbc</code>	<code>\foo æbc\relax</code>	<code>pdflatex</code>	<code><æ><bc></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 63 shows the test. Essentially, the following two expressions are equal under NFSS:

`\@car<test1>\@nil`, where `<test1>` expands to `\IeC {<test1>}`
`\@car<test2>\@nil`, where `<test2>` expands to `\IeC {<test2>}`

If `<test2>` expands to the letter `<test2>`, then it will fail the equality. “Active” characters expand to “two-argument wide” values under NFSS:

<code>\def\{a{<L>}</code>	<code>\protected@edef\{a{<L>}</code>	<code>\protected\edef\{a{<L>}</code>
<code>A macro:->A</code>	<code>A macro:->A</code>	<code>A \protected macro:->A</code>
<code>À macro:->ÃÃ</code>	<code>À macro:->\IeC {\‘A}</code>	<code>À \protected macro:->À</code>
<code>ß macro:->Ã§</code>	<code>ß macro:->\IeC {\ss }</code>	<code>ß \protected macro:->\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.



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

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

Using explicit spacing macros prevents gobbled spaces:

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

See also Sections 2.4.6 and 2.4.7.

2.11.4 L^AT_EX Engines



The `nameauth` package tries to work with multiple languages and typesetting engines. The following preamble snippet from this manual illustrates how that can be done.²⁴

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

This arrangement worked best for this manual, which is tested with all of the L^AT_EX 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
```

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

2.11.5 Hooks: Intro

Margin
Paragraphs



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

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

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

```
\PretagName{Vlad, Tepeş}{Vlad Tepeş}% for accented names

\Name{Vlad III, Dracula}, known as \AKA{Vlad III, Dracula}{Vlad,
Tepeş} (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 Tepeş (the Impaler) after his death, was the son of **Vlad II Dracul**, a member of the Order of the Dragon. Later references to “Vlad III” appear thus.

Now again we have reverted to the original form of `\NamesFormat` and we get Vlad III Dracula and Vlad III. For references to “Vlad” consider using `\Name{Vlad, III}` and use `\NameAddInfo` and `\NameQueryInfo` to handle “Dracula.” The simplified interface greatly helps one to avoid confusion and settle on specific name forms.



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

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

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

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

2.11.6 Hooks: Life Dates

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

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



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

```

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

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

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



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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

Please remember to reset the formatting, if needed:

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

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

2.11.7 Hooks: Advanced

The Name Parser for Hooks

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

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

`\NameParser`

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

```
\renewcommand*{Hook}[1]
  {\cseq_1...cseq_m} \NameParser {cseq_n...}
```

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



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

Continental Format

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

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

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

```
\newif\ifSC
\SCtrue%                               We want small caps in the index
\def\DoFormat#1%
{%
  \ifSC \textsc{#1}%                   Format small caps if true
  \else #1%                           Do nothing if false
  \fi
}
```

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

Since Continental formatting alters the surname, see how the formatting macro always appears in the $\langle SNN \rangle$ field below as we start to define names:

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

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

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

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

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

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

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

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

First	Next	Long	Short
John Quincy ADAMS	Adams	John Quincy Adams	John Quincy
John David ROCKEFELLER III	Rockefeller	John David Rockefeller III	John David
ÆPELSTAN	Æpelstan	Æpelstan	Æpelstan
CHARLES I	Charles	Charles I	Charles
CAO Cao	Cao	Cao Cao	Cao

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

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

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

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

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

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

Caps within Formatting



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

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

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

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

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

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

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

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

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

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

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

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

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

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

First	Next	Long	Short
<code>\deSmet</code>	<code>\deSmet</code>	<code>\LdeSmet</code>	<code>\SdeSmet</code>
Pierre-Jean <i>de Smet</i>	de Smet	Pierre-Jean de Smet	Pierre-Jean

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

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

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

2.11.8 Full Redesign



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

`\NameauthName` 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@FullNamefalse}
\newcommand*\MyFName[3][1=\@empty, 3=\@empty]
{\langle Short name \rangle \@nameauth@FirstNamefalse}
\makeatother
```

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

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

2.12 Naming Pattern Reference

Western Names

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

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

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

Western Plus Affixes

Always use a comma to delimit name/affix pairs.

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

<i>First reference:</i>	<code>\Patton</code>	<code>\Name [George S.]{Patton, Jr.}</code>
George S. Patton Jr.	<code>\LPatton</code>	<code>\Name*[George S.]{Patton, Jr.}</code>
	<code>\SPatton</code>	<code>\FName[George S.]{Patton, Jr.}</code>
<i>Subsequent full:</i>	<code>\LPatton</code>	<code>\Name*[George S.]{Patton, Jr.}</code>
George S. Patton Jr.		
<i>Subsequent surname:</i> Patton	<code>\Patton</code>	<code>\Name [George S.]{Patton, Jr.}</code>
<i>Subsequent forename:</i> George S.	<code>\SPatton</code>	<code>\FName[George S.]{Patton, Jr.}</code>
<i>Alternate forename:</i> George	<code>\SPatton[George]</code>	
	<code>\FName[George S.]{Patton, Jr.}[George]</code>	

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

New Syntax: Non-Western

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

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

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

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

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

Old Syntax: Royal and Eastern

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

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

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

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

3.0



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

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

Particles, American Style

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

<i>First:</i> Walter de la Mare	\DLM	\Name [Walter]{de la Mare}
	\LDLM	\Name*[Walter]{de la Mare}
	\SDLM	\FName[Walter]{de la Mare}
<i>Subsequent:</i> de la Mare	\DLM	\Name [Walter]{de la Mare}
<i>Start of sentence:</i> De la Mare	\CapThis\DLM	\CapThis\Name[Walter]{de la Mare}
<i>Forename:</i> Walter	\SDLM	\FName[Walter]{de la Mare}

Particles, European Style, in the Index

<i>First use:</i>	\JWG [Johann Wolfgang von]	\Name [J.W. von]{Goethe}%
Johann Wolfgang von	\LJWG[Johann Wolfgang von]	[Johann Wolfgang von]
Goethe	\SJWG[Johann Wolfgang von]	\Name*[J.W. von]{Goethe}%
		[Johann Wolfgang von]
		\FName[J.W. von]{Goethe}%
		[Johann Wolfgang von]
<i>Subsequent:</i> Goethe	\JWG	\Name [J.W. von]{Goethe}
<i>Forenames:</i> Johann Wolf-	\SJWG[Johann Wolfgang]	\FName[J.W. von]{Goethe}%
gang		[Johann Wolfgang]

Particles, European Style, Not in the Index

<i>First:</i>	\Harnack [Adolf von]	\Name [Adolf]{Harnack}[Adolf von]
Adolf von Harnack	\LHarnack[Adolf von]	\Name-[Adolf]{Harnack}[Adolf von]
	\SHarnack[Adolf von]	\FName[Adolf]{Harnack}[Adolf von]
<i>Subsequent full:</i>	\LHarnack[Adolf von]	\Name*[Adolf]{Harnack}[Adolf von]
Adolf von Harnack		
<i>Subseq. last:</i> Harnack	\Harnack	\Name [Adolf]{Harnack}
<i>Subseq. first:</i> Adolf	\SHarnack	\FName[Adolf]{Harnack}

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

2.13 Errors and Warnings

Here are some ways to avoid common errors:

- Keep it simple! Avoid unneeded macros and use the simplified interface.
- Check braces and brackets with naming macros to avoid errors like “Paragraph ended...” and “Missing *⟨grouping token⟩* inserted.”
- Do not apply a formatting macro to an entire comma-delimited *⟨SNN, affix⟩* pair. Format each part separately.
- Consider using `\PretagName` with all names containing control sequences or active Unicode; see Section 2.7.5.
- One way to spot errors is to compare index entries with names in the body text. All macros that produce output also emit meaningful warnings.

The older syntax presents its own group of potential errors:

- Erroneously typing `\Name[Henry]{VIII}` prints “Henry VIII” and “VIII,” as well as producing a malformed index entry.
- Avoid forms like `\Name[Henry]{VIII}[Tudor]` which gives “Tudor VIII” and “VIII.” That is a Western alternate name form, which is incorrect.
- The older syntax will not work with some macros. The comma-suffixed form does work with those macros. See Section 2.10.

Warnings result from the following:

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

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

Sometimes dedefinition is harmless because it produces no unwanted results. It is up to the user to consider these warnings.

- Use the `verbose` option for warnings from the indexing macros.
- Using an index cross-reference name as a page entry. Nothing will happen.
- Creating the same cross-reference multiple times. Nothing will happen.
- Creating a page reference after a cross-reference has been created or after you have used `\ExcludeName`. Nothing will happen until you use a variant of `\Includename`.
- Using `\TagName` and `\UntagName` on cross-references. Nothing will happen.
- Using `\PretagName` with cross-references will create sorting tags for them, but also will generate “informational warnings” only if the `verbose` option is selected.
- Using `\ExcludeName` with cross-references. Nothing will happen.
- Using `\ExcludeName` to exclude a name that has already been excluded. Likewise, it will do nothing.

3 Implementation

3.1 Flags and Registers

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

Who Called Me?

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

```
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 2.11.6 and 2.11.7.

```
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 this flag:

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

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

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

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

```
8 \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.

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

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

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

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

```
12 \newif\if@nameauth@FirstFormat
13 \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.

```
14 \newif\if@nameauth@AlwaysComma
15 \newif\if@nameauth@ShowComma
16 \newif\if@nameauth@NoComma
```

Name Breaking

`\KeepAffix` toggles the flag below, which causes an $\langle SNN \rangle$ - $\langle Affix \rangle$ pair to have a non-breaking space between them. For “native” Eastern names this prevents the family name and personal name from breaking.

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

Detect Punctuation

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

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

Long and Short Names

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

```
19 \newif\if@nameauth@FullName
20 \newif\if@nameauth@FirstName
21 \newif\if@nameauth@AltAKA
22 \newif\if@nameauth@OldAKA
23 \newif\if@nameauth@ShortSNN
24 \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.

```
25 \newif\if@nameauth@RevAll
26 \newif\if@nameauth@RevThis
27 \newif\if@nameauth@AllCaps
28 \newif\if@nameauth@AllThis
```

Last-Comma-First

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

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

Capitalize First Letter

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

```
31 \newif\if@nameauth@DoCaps
32 \newif\if@nameauth@UTF
33 \newif\if@nameauth@Accent
```

Warning Levels

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

```
34 \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`.

```
35 \newtoks\@nameauth@toksa%
```

```
36 \newtoks\@nameauth@toksb%
```

```
37 \newtoks\@nameauth@toksc%
```

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

```
38 \newtoks\@nameauth@etoksb%
```

```
39 \newtoks\@nameauth@etoksc%
```

```
40 \newtoks\@nameauth@etoksd%
```

3.2 Hooks

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

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

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

```
42 \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.

```
43 \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.

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

`\NameauthName` Hook to create custom naming macros. Usually the three macros below have the same control sequence, but they need not do so if you want something different. See Section 2.11.8. Use at your own risk! Changing these macros basically rewrites this package.

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

`\NameauthLName` Customization hook called after `\@nameauth@FullName` is set true. See Section 2.11.8.

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

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

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

3.3 Package Options

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

```
48 \DeclareOption{comma}{\@nameauth@AlwaysComma>true}
49 \DeclareOption{nocomma}{\@nameauth@AlwaysComma>false}
50 \DeclareOption{mainmatter}{\@nameauth@MainFormat>true}
51 \DeclareOption{frontmatter}{\@nameauth@MainFormat>false}
52 \DeclareOption{formatAKA}{\@nameauth@AKAFormat>true}
53 \DeclareOption{oldAKA}{\@nameauth@oldAKA>true}
54 \DeclareOption{index}{\@nameauth@DoIndex>true}
55 \DeclareOption{noindex}{\@nameauth@DoIndex>false}
56 \DeclareOption{pretag}{\@nameauth@Pretag>true}
57 \DeclareOption{nopretag}{\@nameauth@Pretag>false}
58 \DeclareOption{allcaps}{\@nameauth@AllCap>true}
59 \DeclareOption{normalcaps}{\@nameauth@AllCaps>false}
60 \DeclareOption{allreversed}%
61   {\@nameauth@RevAll>true\@nameauth@RevAllComma>false}
62 \DeclareOption{allrevcomma}%
63   {\@nameauth@RevAll>true\@nameauth@RevAllComma>true}
64 \DeclareOption{notreversed}%
65   {\@nameauth@RevAll>false\@nameauth@RevAllComma>false}
66 \DeclareOption{alwaysformat}{\@nameauth@AlwaysFormat>true}
67 \DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
68 \DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
69 \DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
70 \DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
71 \DeclareOption{verbose}{\@nameauth@Verboset=true}
72 \ExecuteOptions%
73   {nocomma,%
74     mainmatter,%
75     index,%
76     pretag,%
77     normalcaps,%
78     notreversed,%
79     noformat}
80 \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.

```
81 \RequirePackage{etoolbox}
82 \RequirePackage{suffix}
83 \RequirePackage{trimspaces}
84 \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.

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

Parsing: Root and Suffix

<code>\@nameauth@Root</code>	The following two macros return everything before a comma in $\langle SNN \rangle$. <pre>87 \newcommand*\@nameauth@Root[1]{\@nameauth@TrimRoot#1,\}}</pre>
<code>\@nameauth@TrimRoot</code>	Throw out the comma and suffix, return the radix. <pre>88 \def\@nameauth@TrimRoot#1,#2\{\trim@spaces{#1}}</pre>
<code>\@nameauth@TagRoot</code>	The following two macros return everything before a vertical bar () in an index tag. <pre>89 \newcommand*\@nameauth@TagRoot[1]{\@nameauth@TrimTag#1 \\}</pre>
<code>\@nameauth@TrimTag</code>	Throw out the bar and suffix, return the radix. <pre>90 \def\@nameauth@TrimTag#1 #2\{\#1}</pre>
<code>\@nameauth@Suffix</code>	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. <pre>91 \newcommand*\@nameauth@Suffix[1]{\@nameauth@TrimSuffix#1,,\}</pre>
<code>\@nameauth@TrimSuffix</code>	Throw out the radix; return the suffix with no leading spaces. <pre>92 \def\@nameauth@TrimSuffix#1,#2,#3\{\% 93 \ifx\\#2\\\@empty\else\trim@spaces{#2}\fi}</pre>

Parsing: Capitalization

<code>\@nameauth@AllCapRoot</code>	This macro returns a fully-capitalized radix. It is used for generating capitalized Eastern family names in the body text. It is also the trivial case. <pre>94 \newcommand*\@nameauth@AllCapRoot[1] 95 {\uppercase{\@nameauth@Root{#1}}}</pre>
<code>\@nameauth@UTFtest</code>	Before we attempt at capitalizing anything, we need to determine if we are running under <code>xelatex</code> or <code>lualatex</code> by testing for <code>\Umathchar</code> . 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 <code>\@nameauth@Name</code> and <code>\AKA</code> . <pre>96 \newcommand*\@nameauth@UTFtest[1] 97 {\% 98 \ifdefined\Umathchar 99 \@nameauth@UTFfalse% 100 \else 101 \ifdefined\UTFviii@two@octets 102 \toks@\expandafter{\@car#1\@nil}% 103 \edef\one{\the\toks@}% 104 \toks@\expandafter{\@car\@nil}% 105 \edef\two{\the\toks@}% 106 \ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi 107 \if@nameauth@Accent 108 \@nameauth@UTFtrue% 109 \@nameauth@Accentfalse% 110 \fi 111 \else\@nameauth@UTFfalse\fi 112 \fi 113 }</pre>

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

```
114 \newcommand*\@nameauth@CapRoot[1]
115 {%
116   \if@nameauth@UTF \@nameauth@CRiii#1\\%
117   \else \@nameauth@CRii#1\\%
118   \fi
119 }
```

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

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

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

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

Parsing: Punctuation Detection

`\@nameauth@TestDot` This macro, based on a snippet by Uwe Lueck, checks for a period at the end of its argument. It determines whether we need to call `\@nameauth@CheckDot` below.

```
122 \newcommand*\@nameauth@TestDot[1]
123 {%
124   \def\TestDot##1.\TestEnd##2\\{\TestPunct{##2}}%
125   \def\TestPunct##1{%
126     \ifx\TestPunct##1\TestPunct%
127     \else
128       \@nameauth@Puncttrue%
129     \fi
130   }%
131   \@nameauth@Punctfalse%
132   \TestDot#1\TestEnd.\TestEnd\\%
133 }
```

`\@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.

```
134 \newcommand*\@nameauth@CheckDot%
135 {\futurelet\@token\@nameauth@EvalDot}
```

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

```
136 \newcommand*\@nameauth@EvalDot%
137 {%
138   \let\@period=.%
139   \ifx\@token\@period\expandafter\@gobble \fi
140 }
```

Error Detection

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

```
141 \newcommand*\@nameauth@Error[2]
142 {%
143   \edef\msg{#2}%
144   \edef\pkg{nameauth}%
145   \let\ex\expandafter%
```



```

146 \protected@edef\argb{\trim@spaces{#1}}%
147 \protected@edef\rootb{\@nameauth@Root{#1}}%
148 \ifx\argb\@empty
149 \ex\PackageError\ex{\ex\pkg\ex}\ex{\msg SNN field empty}%
150 \else
151 \ifx\rootb\@empty
152 \ex\PackageError\ex{\ex\pkg\ex}\ex{\msg SNN field malformed}%
153 \fi
154 \fi
155 }

```

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.

```

156 \newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
157 {%

```

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

```

158 \if@nameauth@Lock\else
159 \@nameauth@Locktrue%

```

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

```

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

```

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

```

162 \leavevmode\hbox{}%
163 \IndexName[#1]{#2}[#3]%
164 \if@nameauth@MainFormat
165 \@nameauth@Parse[#1]{#2}[#3]{!MN}%
166 \else
167 \@nameauth@Parse[#1]{#2}[#3]{!NF}%
168 \fi
169 \IndexName[#1]{#2}[#3]%

```

Reset all the “per name” Boolean values.

```

170 \@nameauth@Lockfalse%
171 \@nameauth@InNamefalse%
172 \@nameauth@NBSPfalse%
173 \@nameauth@DoCapsfalse%
174 \@nameauth@Accentfalse%
175 \@nameauth@AllThisfalse%
176 \@nameauth@ShowCommafalse%
177 \@nameauth@NoCommafalse%
178 \@nameauth@RevThisfalse%
179 \@nameauth@RevThisCommafalse%
180 \@nameauth@ShortSNNfalse%
181 \@nameauth@EastFNfalse%

```

Close the “locked” branch.

```

182 \fi

```

Call the full stop detection.

```

183 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
184 }

```

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

```
185 \newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
186 {%
187   \if@nameauth@Lock
```

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

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

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

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

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

```
194   \if@nameauth@DoCaps
195     \@nameauth@UTFtest{#2}%
196     \protected@edef\rootb{\@nameauth@CapRoot{#2}}%
197   \else
198     \if@nameauth@AllThis
199       \protected@edef\rootb{\@nameauth@AllCapRoot{#2}}%
200     \else
201       \protected@edef\rootb{\@nameauth@Root{#2}}%
202     \fi
203   \fi
```

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

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

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

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

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

```
210   \edef\Space{\space}%
211   \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
212   \ifx\arga\@empty\else
213     \if@nameauth@AlwaysComma
214       \edef\Space{,\space}%
215     \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
216   \fi
217   \if@nameauth@ShowComma
218     \edef\Space{,\space}%
219   \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
220   \fi
221   \if@nameauth@NoComma
222     \edef\Space{\space}%
223   \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
224   \fi
225   \fi
```

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

```
226 \let\SNN\rootb%
227 \ifx\arga\@empty
228 \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.

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

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

```
233 \ifx\suffb\@empty
234 \let\FNN\argc%
235 \let\SNN\rootb%
236 \@nameauth@NonWest{\csbc#4}%
237 \else
238 \let\FNN\argc%
239 \let\SNN\rootb%
240 \@nameauth@NonWest{\csb#4}%
241 \fi
242 \fi
243 \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.

```
244 \ifx\argc\@empty
245 \let\FNN\arga%
246 \else
247 \let\FNN\argc%
248 \fi
249 \ifx\suffb\@empty
250 \else
251 \protected@edef\SNN{\rootb\Space\suffb}%
252 \if@nameauth@ShortSNN\let\SNN\rootb\fi
253 \fi
254 \@nameauth@West{\csab#4}%
255 \fi
256 \fi
257 }
```

`\@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.

```
258 \newcommand*\@nameauth@NonWest[1]
259 {%
260 \if@nameauth@Lock
261 \ifcsname#1\endcsname\else
262 \@nameauth@FirstFormattrue%
263 \fi
264 \if@nameauth@InAKA
```

```

265     \if@nameauth@AltAKA
266     \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi
267     \@nameauth@FullNamefalse%
268     \@nameauth@FirstNametrue%
269     \else
270     \@nameauth@FullNametrue%
271     \@nameauth@FirstNamefalse%
272     \fi
273     \else
274     \ifcsname#1\endcsname\else
275     \@nameauth@FullNametrue%
276     \@nameauth@FirstNamefalse%
277     \fi
278     \fi
279     \if@nameauth@FirstName
280     \@nameauth@FullNamefalse%
281     \fi
282     \ifx\FNN\@empty
283     \@nameauth@Hook{\SNN}%
284     \else
285     \if@nameauth@FullName
286     \if@nameauth@RevThis
287     \@nameauth@Hook{\FNN\Space\SNN}%
288     \else
289     \@nameauth@Hook{\SNN\Space\FNN}%
290     \fi
291     \else
292     \if@nameauth@EastFN
293     \@nameauth@Hook{\FNN}%
294     \else
295     \@nameauth@Hook{\SNN}%
296     \fi
297     \fi
298     \fi
299     \ifcsname#1\endcsname\else
300     \if@nameauth@InAKA\else\csgdef{#1}{-}\fi%
301     \fi
302     \@nameauth@FullNamefalse%
303     \@nameauth@FirstNamefalse%
304     \fi
305 }

```

`\@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.

```

306 \newcommand*\@nameauth@West[1]
307 {%
308   \if@nameauth@Lock
309   \ifcsname#1\endcsname\else
310   \@nameauth@FirstFormattrue%
311   \fi
312   \if@nameauth@InAKA
313   \if@nameauth@AltAKA
314   \@nameauth@FullNamefalse%
315   \@nameauth@FirstNametrue%
316   \else

```

```

317         \@nameauth@FullNametrue%
318         \@nameauth@FirstNamefalse%
319     \fi
320 \else
321     \ifcsname#1\endcsname\else
322         \@nameauth@FullNametrue%
323         \@nameauth@FirstNamefalse%
324     \fi
325 \fi
326 \if@nameauth@FirstName
327     \@nameauth@FullNamefalse%
328 \fi
329 \if@nameauth@FullName
330     \if@nameauth@RevThis
331         \@nameauth@Hook{\SNN\space\FNN}%
332     \else
333         \if@nameauth@RevThisComma
334             \edef\RevSpace{,\space}%
335             \@nameauth@Hook{\SNN\RevSpace\FNN}%
336         \else
337             \@nameauth@Hook{\FNN\space\SNN}%
338         \fi
339     \fi
340 \else
341     \if@nameauth@FirstName
342         \@nameauth@Hook{\FNN}%
343     \else
344         \@nameauth@Hook{\rootb}%
345     \fi
346 \fi
347 \ifcsname#1\endcsname\else
348     \if@nameauth@InAKA\else\csgdef{#1}{-}\fi%
349 \fi
350 \@nameauth@FullNamefalse%
351 \@nameauth@FirstNamefalse%
352 \fi
353 }

```

Format Hook Dispatcher

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

```

354 \newcommand*\@nameauth@Hook[1]
355 {%
356     \if@nameauth@Lock
357         \@nameauth@InHooktrue%
358         \protected@edef\test{#1}%
359         \expandafter\@nameauth@TestDot\expandafter{\test}%
360     \if@nameauth@InAKA
361         \if@nameauth@AlwaysFormat
362             \@nameauth@FirstFormattrue%
363         \else
364             \if@nameauth@AKAFormat\else
365                 \@nameauth@FirstFormatfalse\fi

```

```

366     \fi
367     \if@nameauth@MainFormat
368         \if@nameauth@FirstFormat
369             \bgroup\NamesFormat{#1}\egroup%
370         \else
371             \bgroup\MainNameHook{#1}\egroup%
372         \fi
373     \else
374         \if@nameauth@FirstFormat
375             \bgroup\FrontNamesFormat{#1}\egroup%
376         \else
377             \bgroup\FrontNameHook{#1}\egroup%
378         \fi
379     \fi
380 \else
381     \if@nameauth@AlwaysFormat
382         \@nameauth@FirstFormattrue%
383     \fi
384     \if@nameauth@MainFormat
385         \if@nameauth@FirstFormat
386             \bgroup\NamesFormat{#1}\egroup%
387         \else
388             \bgroup\MainNameHook{#1}\egroup%
389         \fi
390     \else
391         \if@nameauth@FirstFormat
392             \bgroup\FrontNamesFormat{#1}\egroup%
393         \else
394             \bgroup\FrontNameHook{#1}\egroup%
395         \fi
396     \fi
397 \fi
398 \@nameauth@FirstFormatfalse%
399 \@nameauth@InHookfalse%
400 \fi
401 }

```

Indexing Internals

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

```

402 \newcommand*\@nameauth@Index[2]
403 {%
404     \def\cseq{#1}%
405     \let\ex\expandafter%
406     \ifcsname\cseq!TAG\endcsname
407         \protected@edef\Tag{\csname#1!TAG\endcsname}%
408         \ex\def\ex\ShortTag\ex{\ex\@nameauth@TagRoot\ex{\Tag}}%
409     \fi
410     \if@nameauth@DoIndex
411         \ifcsname\cseq!TAG\endcsname
412             \ifcsname\cseq!PRE\endcsname
413                 \if@nameauth@Xref%
414                     \index%
415                     {\csname\cseq!PRE\endcsname#2\ShortTag}%
416                 \else
417                     \index%

```

```

418         {\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
419         \fi
420     \else
421         \if@nameauth@Xref
422             \index{#2\ShortTag}%
423         \else
424             \index{#2\csname\cseq!TAG\endcsname}%
425         \fi
426     \fi
427 \else
428     \ifcsname\cseq!PRE\endcsname
429         \index{\csname\cseq!PRE\endcsname#2}%
430     \else
431         \index{#2}%
432     \fi
433 \fi
434 \fi
435 }

```

`\@nameauth@Actual` This sets the “actual” character used by `nameauth` for index sorting.

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

3.5 User Interface Macros

Syntactic Formatting — Capitalization

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

```
437 \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.

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

`\CapName` Capitalize entire required name.

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

`\AllCapsInactive` Turn off global surname capitalization.

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

`\AllCapsActive` Turn on global surname capitalization.

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

Syntactic Formatting — Reversing

`\RevName` Reverse name order.

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

`\ReverseInactive` Turn off global name reversing.

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

`\ReverseActive` Turn on global name reversing.

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

`\ForceFN` Force the printing of an Eastern forename in the text.
 446 `\newcommand*\ForceFN{\@nameauth@EastFNtrue}`

Syntactic Formatting — Reversing with Commas

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

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

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

Syntactic Formatting — Affixes

`\ShowComma` Put comma between name and suffix one time.
 453 `\newcommand*\ShowComma{\@nameauth@ShowCommatrue}`

`\NoComma` Remove comma between name and suffix one time (with comma option).
 454 `\newcommand*\NoComma{\@nameauth@NoCommatrue}`

`\DropAffix` Suppress the affix in a long Western name.
 455 `\newcommand*\DropAffix{\@nameauth@ShortSNNtrue}`

Typographic Formatting — Affixes

`\KeepAffix` Trigger a name-suffix pair to be separated by a non-breaking space.
 456 `\newcommand*\KeepAffix{\@nameauth@NBSPtrue}`

Typographic Formatting — Main Versus Front Matter

`\NamesInactive` Switch to the “non-formatted” species of names.
 457 `\newcommand*\NamesInactive{\@nameauth@MainFormatfalse}`

`\NamesActive` Switch to the “formatted” species of names.
 458 `\newcommand*\NamesActive{\@nameauth@MainFormattrue}`

Name Occurrence Tweaks

`\LocalNames` `\LocalNames` sets `@nameauth@LocalNames` true so `\ForgetName` and `\SubvertName` do not affect both formatted and unformatted names.
 459 `\newcommand*\LocalNames{\global\@nameauth@LocalNamestrue}`

`\GlobalNames` `\GlobalNames` sets `@nameauth@LocalNames` false, restoring the default behavior of `\ForgetName` and `\SubvertName`.
 460 `\newcommand*\GlobalNames{\global\@nameauth@LocalNamesfalse}`

Index Operations

`\IndexInactive` turn off global indexing of names.
 461 `\newcommand*\IndexInactive{\@nameauth@DoIndexfalse}`

`\IndexActive` turn on global indexing of names.

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

`\IndexActual` Change the “actual” character from the default.

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

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

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

Hook Macro Name Parser

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

```
466 \newcommand*\NameParser
467 {%
468   \if@nameauth@InHook
469     \let\SNN\rootb%
470     \ifx\arga\@empty
471       \ifx\argc\@empty
472         \let\FNN\suffb%
473       \else
474         \let\FNN\argc%
475       \fi
476     \ifx\suffb\@empty
477       \SNN%
478     \else
479       \if@nameauth@FullName
480         \if@nameauth@RevThis
481           \FNN\Space\SNN%
482         \else
483           \SNN\Space\FNN%
484         \fi
485       \else
486         \if@nameauth@EastFN
487           \FNN%
488         \else
489           \SNN%
490         \fi
491       \fi
492     \fi
493   \else
494     \ifx\argc\@empty
495       \let\FNN\arga%
496     \else
497       \let\FNN\argc%
498     \fi
499     \ifx\suffb\@empty
500     \else
501       \protected@edef\SNN{\rootb\Space\suffb}%
502       \if@nameauth@ShortSNN\let\SNN\rootb\fi%
503     \fi
504     \if@nameauth@FullName
505     \if@nameauth@RevThis
```

```

506         \SNN\space\FNN%
507     \else
508         \if@nameauth@RevThisComma
509             \SNN\RevSpace\FNN%
510         \else
511             \FNN\space\SNN%
512         \fi
513     \fi
514 \else
515     \if@nameauth@FirstName
516         \FNN%
517     \else
518         \protected@edef\SNN{\rootb}%
519         \SNN%
520     \fi
521 \fi
522 \fi
523 \fi
524 }

```

Main Naming Interface

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

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

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

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

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

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

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

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

Index Operations

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

```

531 \newcommandx*\IndexName[3][1=\@empty, 3=\@empty]
532 {%
533     \protected@edef\testa{#1}%
534     \protected@edef\arga{\trim@spaces{#1}}%
535     \protected@edef\testb{\trim@spaces{#2}}%
536     \protected@edef\rootb{\@nameauth@Root{#2}}%
537     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
538     \protected@edef\testc{#3}%
539     \protected@edef\argc{\trim@spaces{#3}}%
540     \def\csb{\@nameauth@Clean{#2}}%
541     \def\csbc{\@nameauth@Clean{#2,#3}}%
542     \def\csab{\@nameauth@Clean{#1!#2}}%

```

Test for malformed input.

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

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

```

544 \let\Short\rootb%
545 \ifx\suffb\@empty
546 \let\SNN\rootb%
547 \else
548 \protected@edef\SNN{\rootb\space\suffb}%
549 \fi

```

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

```

550 \ifx\testa\@empty
551 \ifx\testc\@empty
552 \ifcsname\csb!PN\endcsname
553 \if@nameauth@Verbose
554 \PackageWarning{nameauth}%
555 {macro \IndexName: XRef: #2 exists}%
556 \fi
557 \else
558 \@nameauth@Index{\csb}{\SNN}%
559 \fi
560 \else
561 \ifx\suffb\@empty
562 \ifcsname\csbc!PN\endcsname
563 \if@nameauth@Verbose
564 \PackageWarning{nameauth}%
565 {macro \IndexName: XRef: #2 #3 exists}%
566 \fi
567 \else
568 \@nameauth@Index{\csbc}{\SNN\space\argc}%
569 \fi
570 \else
571 \ifcsname\csb!PN\endcsname
572 \if@nameauth@Verbose
573 \PackageWarning{nameauth}%
574 {macro \IndexName: XRef: #2 exists}%
575 \fi
576 \else
577 \@nameauth@Index{\csb}{\SNN}%
578 \fi
579 \fi
580 \fi
581 \else
582 \ifcsname\csab!PN\endcsname
583 \if@nameauth@Verbose
584 \PackageWarning{nameauth}%
585 {macro \IndexName: XRef: #1 #2 exists}%
586 \fi
587 \else
588 \ifx\suffb\@empty
589 \@nameauth@Index{\csab}%
590 {\Short,\space\arga}%
591 \else
592 \@nameauth@Index{\csab}%
593 {\Short,\space\arga,\space\suffb}%
594 \fi

```

```

595 \fi
596 \fi
597 }

```

`\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.

```

598 \newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
599 {%
600 \protected@edef\testa{#1}%
601 \protected@edef\arga{\trim@spaces{#1}}%
602 \protected@edef\rootb{\@nameauth@Root{#2}}%
603 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
604 \protected@edef\testc{#3}%
605 \protected@edef\argc{\trim@spaces{#3}}%
606 \protected@edef\target{#4}%
607 \def\csb{\@nameauth@Clean{#2}}%
608 \def\csbc{\@nameauth@Clean{#2,#3}}%
609 \def\csab{\@nameauth@Clean{#1!#2}}%
610 \let\ex\expandafter%

```

Test for malformed input.

```

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

```

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

```

613 \let\Short\rootb%
614 \ifx\suffb\@empty
615 \let\SNN\rootb%
616 \else
617 \protected@edef\SNN{\rootb\space\suffb}%
618 \fi

```

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

```

619 \ifx\testa\@empty
620 \ifx\testc\@empty
621 \ifcsname\csb!PN\endcsname
622 \if@nameauth@Verbose
623 \PackageWarning{nameauth}%
624 {macro \IndexRef: XRef: #2 exists}%
625 \fi
626 \else
627 \if@nameauth@SeeAlso
628 \@nameauth@Index{\csb}{\SNN|seealso{\target}}%
629 \else
630 \@nameauth@Index{\csb}{\SNN|see{\target}}%
631 \fi
632 \csgdef{\csb!PN}{}%
633 \fi
634 \else
635 \ifx\suffb\@empty
636 \ifcsname\csbc!PN\endcsname
637 \if@nameauth@Verbose
638 \PackageWarning{nameauth}%
639 {macro \IndexRef: XRef: #2 #3 exists}%
640 \fi
641 \else

```

```

642         \if@nameauth@SeeAlso
643         \@nameauth@Index{\csbc}%
644         {\SNN\space\argc|seealso{\target}}}%
645     \else
646         \@nameauth@Index{\csbc}%
647         {\SNN\space\argc|see{\target}}}%
648     \fi
649     \csgdef{\csbc!PN}{}%
650 \fi
651 \else
652     \ifcsname\csb!PN\endcsname
653     \if@nameauth@Verbose
654     \PackageWarning{nameauth}%
655     {macro \IndexRef: XRef: #2 exists}%
656     \fi
657     \else
658     \if@nameauth@SeeAlso
659     \@nameauth@Index{\csb}%
660     {\SNN|seealso{\target}}}%
661     \else
662     \@nameauth@Index{\csb}%
663     {\SNN|see{\target}}}%
664     \fi
665     \csgdef{\csb!PN}{}%
666 \fi
667 \fi
668 \fi
669 \else
670     \ifcsname\csab!PN\endcsname
671     \if@nameauth@Verbose
672     \PackageWarning{nameauth}%
673     {macro \IndexRef: XRef: #1 #2 exists}%
674     \fi
675     \else
676     \ifx\suffb\@empty
677     \if@nameauth@SeeAlso
678     \@nameauth@Index{\csab}%
679     {\Short,\space\argc|seealso{\target}}}%
680     \else
681     \@nameauth@Index{\csab}%
682     {\Short,\space\argc|see{\target}}}%
683     \fi
684     \else
685     \if@nameauth@SeeAlso
686     \@nameauth@Index{\csab}%
687     {\Short,\space\argc,\space\suffb|seealso{\target}}}%
688     \else
689     \@nameauth@Index{\csab}%
690     {\Short,\space\argc,\space\suffb|see{\target}}}%
691     \fi
692     \fi
693     \csgdef{\csab!PN}{}%
694 \fi
695 \fi
696 \@nameauth@SeeAlsofalse%
697 \@nameauth@Xreffalse%

```

698 }

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

```
699 \newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]
700 {%
701   \protected@edef\testa{#1}%
702   \protected@edef\testc{#3}%
703   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
704   \def\csb{\@nameauth@Clean{#2}}%
705   \def\csbc{\@nameauth@Clean{#2,#3}}%
706   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
707   \@nameauth@Error{#2}{macro \string\ExcludeName}%
708   \ifx\testa\@empty
709     \ifx\testc\@empty
710       \if@nameauth@Verbose
711         \ifcsname\csb!MN\endcsname
712           \PackageWarning{nameauth}%
713             {macro \ExcludeName: Reference: #2 exists}%
714         \fi
715         \ifcsname\csb!NF\endcsname
716           \PackageWarning{nameauth}%
717             {macro \ExcludeName: Reference: #2 exists}%
718         \fi
719       \fi
720       \ifcsname\csb!PN\endcsname
721         \if@nameauth@Verbose
722           \PackageWarning{nameauth}%
723             {macro \ExcludeName: Xref: #2 exists}%
724         \fi
725       \else
726         \csgdef{\csb!PN}{!}%
727       \fi
728     \else
729       \ifx\suffb\@empty
730         \if@nameauth@Verbose
731           \ifcsname\csbc!MN\endcsname
732             \PackageWarning{nameauth}%
733               {macro \ExcludeName: Reference: #2 #3 exists}%
734           \fi
735           \ifcsname\csbc!NF\endcsname
736             \PackageWarning{nameauth}%
737               {macro \ExcludeName: Reference: #2 #3 exists}%
738           \fi
739         \fi
740         \csgdef{\csbc!PN}{!}%
741         \ifcsname\csbc!PN\endcsname
742           \if@nameauth@Verbose
743             \PackageWarning{nameauth}%
744               {macro \ExcludeName: Xref: #2 exists}%
745           \fi
746         \else
747           \csgdef{\csbc!PN}{!}%
748         \fi
749       \else
```

```

750     \if@nameauth@Verbose
751     \ifcsname\csb!MN\endcsname
752     \PackageWarning{nameauth}%
753     {macro \ExcludeName: Reference: #2 exists}%
754     \fi
755     \ifcsname\csb!NF\endcsname
756     \PackageWarning{nameauth}%
757     {macro \ExcludeName: Reference: #2 exists}%
758     \fi
759     \fi
760     \ifcsname\csb!PN\endcsname
761     \if@nameauth@Verbose
762     \PackageWarning{nameauth}%
763     {macro \ExcludeName: Xref: #2 exists}%
764     \fi
765     \else
766     \csgdef{\csb!PN}{!}%
767     \fi
768     \fi
769     \fi
770   \else
771     \if@nameauth@Verbose
772     \ifcsname\csab!MN\endcsname
773     \PackageWarning{nameauth}%
774     {macro \ExcludeName: Reference: #1 #2 exists}%
775     \fi
776     \ifcsname\csab!NF\endcsname
777     \PackageWarning{nameauth}%
778     {macro \ExcludeName: Reference: #1 #2 exists}%
779     \fi
780     \fi
781     \ifcsname\csab!PN\endcsname
782     \if@nameauth@Verbose
783     \PackageWarning{nameauth}%
784     {macro \ExcludeName: Xref: #2 exists}%
785     \fi
786     \else
787     \csgdef{\csab!PN}{!}%
788     \fi
789     \fi
790 }

```

\IncludeName This macro allows a name to be indexed.

```

791 \newcommandx*\IncludeName[3][1=\@empty, 3=\@empty]
792 {%
793   \protected@edef\testa{#1}%
794   \protected@edef\testc{#3}%
795   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
796   \def\csb{\@nameauth@Clean{#2}}%
797   \def\csbc{\@nameauth@Clean{#2,#3}}%
798   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

799   \@nameauth@Error{#2}{macro \string\IncludeName}%
800   \ifx\testa\@empty

```

```

801 \ifx\testc\@empty
802 \ifcsname\csb!PN\endcsname
803 \edef\testex{\csname\csb!PN\endcsname}%
804 \ifx\testex\@empty\else\global\csundef{\csb!PN}\fi
805 \fi
806 \else
807 \ifx\suffb\@empty
808 \ifcsname\csbc!PN\endcsname
809 \edef\testex{\csname\csbc!PN\endcsname}%
810 \ifx\testex\@empty\else\global\csundef{\csbc!PN}\fi
811 \fi
812 \else
813 \ifcsname\csb!PN\endcsname
814 \edef\testex{\csname\csb!PN\endcsname}%
815 \ifx\testex\@empty\else\global\csundef{\csb!PN}\fi
816 \fi
817 \fi
818 \fi
819 \else
820 \ifcsname\csab!PN\endcsname
821 \edef\testex{\csname\csab!PN\endcsname}%
822 \ifx\testex\@empty\else\global\csundef{\csab!PN}\fi
823 \fi
824 \fi
825 }

```

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

```

826 \WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
827 {%
828 \protected@edef\testa{#1}%
829 \protected@edef\testc{#3}%
830 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
831 \def\csb{\@nameauth@Clean{#2}}%
832 \def\csbc{\@nameauth@Clean{#2,#3}}%
833 \def\csab{\@nameauth@Clean{#1!#2}}%
834 \edef\testa{!}%

```

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

```

835 \@nameauth@Error{#2}{macro \string\IncludeName*}%
836 \ifx\testa\@empty
837 \ifx\testc\@empty
838 \global\csundef{\csb!PN}%
839 \else
840 \ifx\suffb\@empty
841 \global\csundef{\csbc!PN}%
842 \else
843 \global\csundef{\csb!PN}%
844 \fi
845 \fi
846 \else
847 \global\csundef{\csab!PN}%
848 \fi
849 }

```


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

```

850 \newcommandx*\PretagName[4][1=\@empty, 3=\@empty]
851 {%
852   \protected@edef\testa{#1}%
853   \protected@edef\testc{#3}%
854   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
855   \def\csb{\@nameauth@Clean{#2}}%
856   \def\csbc{\@nameauth@Clean{#2,#3}}%
857   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

858   \@nameauth@Error{#2}{\macro \string\PretagName}%
859   \ifx\testa\@empty
860     \ifx\testc\@empty
861       \ifcsname\csb!PN\endcsname
862         \if@nameauth@Verbose
863           \PackageWarning{nameauth}%
864             {macro \PretagName: tagging xref: #2}%
865         \fi
866       \fi
867       \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
868     \else
869       \ifx\suffb\@empty
870         \ifcsname\csbc!PN\endcsname
871           \if@nameauth@Verbose
872             \PackageWarning{nameauth}%
873               {macro \PretagName: tagging xref: #2 #3}%
874           \fi
875         \fi
876         \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
877       \else
878         \ifcsname\csab!PN\endcsname
879           \if@nameauth@Verbose
880             \PackageWarning{nameauth}%
881               {macro \PretagName: tagging xref: #2}%
882           \fi
883         \fi
884         \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
885       \fi
886     \fi
887   \else
888     \ifcsname\csab!PN\endcsname
889       \if@nameauth@Verbose
890         \PackageWarning{nameauth}%
891           {macro \PretagName: tagging xref: #1 #2}%
892       \fi
893     \fi
894     \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
895   \fi
896 }

```

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

```

897 \newcommandx*\TagName[4][1=\@empty, 3=\@empty]
898 {%
899   \protected@edef\testa{#1}%
900   \protected@edef\testc{#3}%
901   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
902   \def\csb{\@nameauth@Clean{#2}}%
903   \def\csbc{\@nameauth@Clean{#2,#3}}%
904   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

905   \@nameauth@Error{#2}{macro \string\TagName}%
906   \ifx\testa\@empty
907     \ifx\testc\@empty
908       \ifcsname\csb!PN\endcsname
909         \if@nameauth@Verbose
910           \PackageWarning{nameauth}%
911             {macro \TagName: not tagging xref: #2}%
912         \fi
913       \else
914         \csgdef{\csb!TAG}{#4}%
915       \fi
916     \else
917       \ifx\suffb\@empty
918         \ifcsname\csbc!PN\endcsname
919           \if@nameauth@Verbose
920             \PackageWarning{nameauth}%
921               {macro \TagName: not tagging xref: #2 #3}%
922           \fi
923         \else
924           \csgdef{\csbc!TAG}{#4}%
925         \fi
926       \else
927         \ifcsname\csb!PN\endcsname
928           \if@nameauth@Verbose
929             \PackageWarning{nameauth}%
930               {macro \TagName: not tagging xref: #2}%
931           \fi
932         \else
933           \csgdef{\csb!TAG}{#4}%
934         \fi
935       \fi
936     \fi
937   \else
938     \ifcsname\csab!PN\endcsname
939       \if@nameauth@Verbose
940         \PackageWarning{nameauth}%
941           {macro \TagName: not tagging xref: #1 #2}%
942       \fi
943     \else
944       \csgdef{\csab!TAG}{#4}%
945     \fi
946   \fi
947 }

```

`\UntagName` This deletes an index tag.

```
948 \newcommandx*\UntagName[3][1=\@empty, 3=\@empty]
949 {%
950   \protected@edef\testa{#1}%
951   \protected@edef\testc{#3}%
952   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
953   \def\csb{\@nameauth@Clean{#2}}%
954   \def\csbc{\@nameauth@Clean{#2,#3}}%
955   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
956   \@nameauth@Error{#2}{macro \string\UntagName}%
957   \ifx\testa\@empty
958     \ifx\testc\@empty
959       \global\csundef{\csb!TAG}%
960     \else
961       \ifx\suffb\@empty
962         \global\csundef{\csbc!TAG}%
963       \else
964         \global\csundef{\csb!TAG}%
965       \fi
966     \fi
967   \else
968     \global\csundef{\csab!TAG}%
969   \fi
970 }
```

Name Info Database: “Text Tags”

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

```
971 \newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
972 {%
973   \protected@edef\testa{#1}%
974   \protected@edef\testc{#3}%
975   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
976   \def\csb{\@nameauth@Clean{#2}}%
977   \def\csbc{\@nameauth@Clean{#2,#3}}%
978   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
979   \@nameauth@Error{#2}{macro \string\NameAddInfo}%
980   \ifx\testa\@empty
981     \ifx\testc\@empty
982       \csgdef{\csb!DB}{#4}%
983     \else
984       \ifx\Suff\@empty
985         \csgdef{\csbc!DB}{#4}%
986       \else
987         \csgdef{\csb!DB}{#4}%
988       \fi
989     \fi
990   \else
991     \csgdef{\csab!DB}{#4}%
992   \fi
993 }
```

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

```

994 \newcommandx\NameQueryInfo[3][1=\@empty, 3=\@empty]
995 {%
996   \protected@edef\testa{#1}%
997   \protected@edef\testc{#3}%
998   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
999   \def\csb{\@nameauth@Clean{#2}}%
1000   \def\csbc{\@nameauth@Clean{#2,#3}}%
1001   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1002 \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
1003 \ifx\testa\@empty
1004   \ifx\testc\@empty
1005     \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1006   \else
1007     \ifx\Suff\@empty
1008       \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
1009     \else
1010       \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1011     \fi
1012   \fi
1013 \else
1014   \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
1015 \fi
1016 }

```

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

```

1017 \newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
1018 {%
1019   \protected@edef\testa{#1}%
1020   \protected@edef\testc{#3}%
1021   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1022   \def\csb{\@nameauth@Clean{#2}}%
1023   \def\csbc{\@nameauth@Clean{#2,#3}}%
1024   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1025 \@nameauth@Error{#2}{macro \string\NameClearInfo}%
1026 \ifx\testa\@empty
1027   \ifx\testc\@empty
1028     \global\csundef{\csb!DB}%
1029   \else
1030     \ifx\Suff\@empty
1031       \global\csundef{\csbc!DB}%
1032     \else
1033       \global\csundef{\csb!DB}%
1034     \fi
1035   \fi
1036 \else
1037   \global\csundef{\csab!DB}%
1038 \fi
1039 }

```

Name Decisions

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

```

1040 \newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
1041 {%
1042   \protected@edef\testa{#1}%
1043   \protected@edef\testc{#3}%
1044   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1045   \def\csb{\@nameauth@Clean{#2}}%
1046   \def\csbc{\@nameauth@Clean{#2,#3}}%
1047   \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and choose the path.

```

1048   \@nameauth@Error{#2}{macro \string\IfMainName}%
1049   \ifx\testa\@empty
1050     \ifx\testc\@empty
1051       \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1052     \else
1053       \ifx\suffb\@empty
1054         \ifcsname\csbc!MN\endcsname{#4}\else{#5}\fi
1055       \else
1056         \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1057       \fi
1058     \fi
1059   \else
1060     \ifcsname\csab!MN\endcsname{#4}\else{#5}\fi
1061   \fi
1062 }

```

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

```

1063 \newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
1064 {%
1065   \protected@edef\testa{#1}%
1066   \protected@edef\testc{#3}%
1067   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1068   \def\csb{\@nameauth@Clean{#2}}%
1069   \def\csbc{\@nameauth@Clean{#2,#3}}%
1070   \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and choose the path.

```

1071   \@nameauth@Error{#2}{macro \string\IfFrontName}%
1072   \ifx\testa\@empty
1073     \ifx\testc\@empty
1074       \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1075     \else
1076       \ifx\suffb\@empty
1077         \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
1078       \else
1079         \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1080       \fi
1081     \fi
1082   \else
1083     \ifcsname\csab!NF\endcsname{#4}\else{#5}\fi
1084   \fi
1085 }

```

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

```

1086 \newcommandx\IfAKA[6][1=\@empty, 3=\@empty]
1087 {%
1088   \protected@edef\testa{#1}%
1089   \protected@edef\testc{#3}%
1090   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1091   \def\csb{\@nameauth@Clean{#2}}%
1092   \def\csbc{\@nameauth@Clean{#2,#3}}%
1093   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1094   \@nameauth@Error{#2}{macro \string\IfAKA}%
1095   \ifx\testa\@empty
1096     \ifx\testc\@empty
1097       \ifcsname\csb!PN\endcsname
1098         \edef\testex{\csname\csb!PN\endcsname}%
1099         \ifx\testex\@empty{#4}\else{#6}\fi
1100       \else{#5}\fi
1101     \else
1102       \ifx\suffb\@empty
1103         \ifcsname\csbc!PN\endcsname
1104           \edef\testex{\csname\csbc!PN\endcsname}%
1105           \ifx\testex\@empty{#4}\else{#6}\fi
1106         \else{#5}\fi
1107       \else
1108         \ifcsname\csb!PN\endcsname
1109           \edef\testex{\csname\csb!PN\endcsname}%
1110           \ifx\testex\@empty{#4}\else{#6}\fi
1111         \else{#5}\fi
1112       \fi
1113     \fi
1114   \else
1115     \ifcsname\csab!PN\endcsname
1116       \edef\testex{\csname\csab!PN\endcsname}%
1117       \ifx\testex\@empty{#4}\else{#6}\fi
1118     \else{#5}\fi
1119   \fi
1120 }

```

Changing Name Decisions

\ForgetName This undefines a control sequence to force the “first use” option of **\Name**.

```

1121 \newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]
1122 {%
1123   \protected@edef\testa{#1}%
1124   \protected@edef\testc{#3}%
1125   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1126   \def\csb{\@nameauth@Clean{#2}}%
1127   \def\csbc{\@nameauth@Clean{#2,#3}}%
1128   \def\csab{\@nameauth@Clean{#1!#2}}%
1129   \@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**).

```

1130   \ifx\testa\@empty

```

```

1131 \ifx\testc\@empty
1132 \if@nameauth@LocalNames
1133 \if@nameauth@MainFormat
1134 \global\csundef{\csb!MN}%
1135 \else
1136 \global\csundef{\csb!NF}%
1137 \fi
1138 \else
1139 \global\csundef{\csb!MN}%
1140 \global\csundef{\csb!NF}%
1141 \fi
1142 \else
1143 \ifx\suffb\@empty
1144 \if@nameauth@LocalNames
1145 \if@nameauth@MainFormat
1146 \global\csundef{\csbc!MN}%
1147 \else
1148 \global\csundef{\csbc!NF}%
1149 \fi
1150 \else
1151 \global\csundef{\csbc!MN}%
1152 \global\csundef{\csbc!NF}%
1153 \fi
1154 \else
1155 \if@nameauth@LocalNames
1156 \if@nameauth@MainFormat
1157 \global\csundef{\csb!MN}%
1158 \else
1159 \global\csundef{\csb!NF}%
1160 \fi
1161 \else
1162 \global\csundef{\csb!MN}%
1163 \global\csundef{\csb!NF}%
1164 \fi
1165 \fi
1166 \fi
1167 \else
1168 \if@nameauth@LocalNames
1169 \if@nameauth@MainFormat
1170 \global\csundef{\csab!MN}%
1171 \else
1172 \global\csundef{\csab!NF}%
1173 \fi
1174 \else
1175 \global\csundef{\csab!MN}%
1176 \global\csundef{\csab!NF}%
1177 \fi
1178 \fi
1179 }

```

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

```

1180 \newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
1181 {%
1182 \protected@edef\testa{#1}%
1183 \protected@edef\testc{#3}%
1184 \protected@edef\suffb{\@nameauth@Suffix{#2}}%

```

```

1185 \def\csb{\@nameauth@Clean{#2}}%
1186 \def\csbc{\@nameauth@Clean{#2,#3}}%
1187 \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them.

```

1188 \@nameauth@Error{#2}{macro \string\SubvertName}%

```

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

```

1189 \ifx\testa\@empty
1190   \ifx\testc\@empty
1191     \if@nameauth@LocalNames
1192       \if@nameauth@MainFormat
1193         \csgdef{\csb!MN}{}%
1194       \else
1195         \csgdef{\csb!NF}{}%
1196       \fi
1197     \else
1198       \csgdef{\csb!MN}{}%
1199       \csgdef{\csb!NF}{}%
1200     \fi
1201   \else
1202     \ifx\suffb\@empty
1203       \if@nameauth@LocalNames
1204         \if@nameauth@MainFormat
1205           \csgdef{\csbc!MN}{}%
1206         \else
1207           \csgdef{\csbc!NF}{}%
1208         \fi
1209       \else
1210         \csgdef{\csbc!MN}{}%
1211         \csgdef{\csbc!NF}{}%
1212       \fi
1213     \fi
1214   \else
1215     \if@nameauth@LocalNames
1216       \if@nameauth@MainFormat
1217         \csgdef{\csb!MN}{}%
1218       \else
1219         \csgdef{\csb!NF}{}%
1220       \fi
1221     \else
1222       \csgdef{\csb!MN}{}%
1223       \csgdef{\csb!NF}{}%
1224     \fi
1225   \fi
1226 \else
1227   \if@nameauth@LocalNames
1228     \if@nameauth@MainFormat
1229       \csgdef{\csab!MN}{}%
1230     \else
1231       \csgdef{\csab!NF}{}%
1232     \fi
1233   \else
1234     \csgdef{\csab!MN}{}%
1235     \csgdef{\csab!NF}{}%

```



```

1236     \fi
1237 \fi
1238 }

```

Alternate Names

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

```

1239 \newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]
1240 {%

```

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

```

1241 \if@nameauth@Lock\else
1242 \@nameauth@Locktrue%

```

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

```

1243 \@nameauth@InAKAtrue%
1244 \@nameauth@Error{#2}{macro \string\AKA}%
1245 \@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.

```

1246 \leavevmode\hbox{}%
1247 \protected@edef\testi{#1}%
1248 \protected@edef\argi{\trim@spaces{#1}}%
1249 \protected@edef\rooti{\@nameauth@Root{#2}}%
1250 \protected@edef\suffi{\@nameauth@Suffix{#2}}%
1251 \@nameauth@Parse[#3]{#4}{#5}{!PN}%

```

Create an index cross-reference based on the arguments.

```

1252 \ifx\testi\@empty
1253   \ifx\suffi\@empty
1254     \IndexRef[#3]{#4}{#5}{\rooti}%
1255   \else
1256     \IndexRef[#3]{#4}{#5}{\rooti\space\suffi}%
1257   \fi
1258 \else
1259   \ifx\suffi\@empty
1260     \IndexRef[#3]{#4}{#5}{\rooti,\space\argi}%
1261   \else
1262     \IndexRef[#3]{#4}{#5}{\rooti,\space\argi,\space\suffi}%
1263   \fi
1264 \fi

```

Reset all the “per name” Boolean values.

```

1265 \@nameauth@Lockfalse%
1266 \@nameauth@InAKAfalse%
1267 \@nameauth@AltAKAfalse%
1268 \@nameauth@NBSPfalse%
1269 \@nameauth@DoCapsfalse%
1270 \@nameauth@Accentfalse%
1271 \@nameauth@AllThisfalse%
1272 \@nameauth@ShowCommafalse%
1273 \@nameauth@NoCommafalse%
1274 \@nameauth@RevThisfalse%
1275 \@nameauth@RevThisCommafalse%
1276 \@nameauth@ShortSNNfalse%
1277 \@nameauth@EastFNfalse%

```

Close the “locked” branch.

```
1278 \fi
```

Call the full stop detection.

```
1279 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
1280 }
```

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

```
1281 \WithSuffix{\newcommand*}\AKA*{\@nameauth@AltAKAtrue\AKA}
```

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

```
1282 \newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]
1283 {\NameauthName[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])}
```

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

```
1284 \WithSuffix{\newcommand*}\PName*{\@nameauth@FullNametrue\PName}
```

Simplified Interface

nameauth The **nameauth** environment creates macro shorthands.

```
1285 \newenvironment{nameauth}{%
1286 \begingroup%
1287 \let\ex\expandafter%
1288 \csdef{<}&##1&##2&##3&##4>{%
1289 \protected@edef\@arga{\trim@spaces{##1}}%
1290 \protected@edef\@testb{\trim@spaces{##2}}%
1291 \protected@edef\@testd{\trim@spaces{##4}}%
1292 \@nameauth@etoks b\expandafter{##2}%
1293 \@nameauth@etoks c\expandafter{##3}%
1294 \@nameauth@etoks d\expandafter{##4}%
1295 \ifx\@arga\@empty
1296 \PackageError{nameauth}%
1297 {environment nameauth: Control sequence missing}%
1298 \fi
1299 \@nameauth@Error{##3}{environment nameauth}%
1300 \ifcsname\@arga\endcsname
1301 \PackageWarning{nameauth}%
1302 {environment nameauth: Shorthand macro already exists}%
1303 \fi
1304 \ifx\@testd\@empty
1305 \ifx\@testb\@empty
1306 \ex\csgdef\ex{\ex\@arga\ex}\ex{\ex\NameauthName\ex{%
1307 \the\@nameauth@etoks c}}%
1308 \ex\csgdef\ex{\ex L\ex\@arga\ex}\ex{%
1309 \ex\@nameauth@FullNametrue%
1310 \ex\NameauthLName\ex{\the\@nameauth@etoks c}}%
1311 \ex\csgdef\ex{\ex S\ex\@arga\ex}\ex{%
1312 \ex\@nameauth@FirstNametrue%
1313 \ex\NameauthFName\ex{\the\@nameauth@etoks c}}%
1314 \else
1315 \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1316 \ex\ex\ex{\ex\ex\ex\NameauthName\ex\ex\ex[%
1317 \ex\the\ex\@nameauth@etoks b\ex]\ex{\the\@nameauth@etoks c}}%
1318 \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
```

```

1319         \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1320         \ex\ex\ex\NameauthLName\ex\ex\ex[%
1321         \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
1322     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
1323     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1324     \ex\ex\ex\NameauthFName\ex\ex\ex[%
1325     \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
1326 \fi
1327 \else
1328     \ifx\@testb\@empty
1329         \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1330         \ex\ex\ex{\ex\ex\ex\NameauthName\ex\ex\ex}%
1331         \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd}}%
1332     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
1333     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1334     \ex\ex\ex\NameauthLName\ex\ex\ex}%
1335     \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd}}%
1336     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
1337     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1338     \ex\ex\ex\NameauthFName\ex\ex\ex}%
1339     \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd}}%
1340 \else
1341     \ex\ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1342     \ex\ex\ex\ex\ex\ex\ex\ex\@arga\ex\ex\ex\ex\ex\ex\ex}%
1343     \ex\ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\ex\NameauthName%
1344     \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb%
1345     \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1346     \the\@nameauth@etoksd}}%
1347     \ex\ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1348     \ex\ex\ex\ex\ex\ex\ex\ex L\ex\ex\ex\ex\ex\ex\ex\ex\@arga%
1349     \ex\ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex\ex{%
1350     \ex\ex\ex\ex\ex\ex\ex\ex\@nameauth@FullNametrue%
1351     \ex\ex\ex\ex\ex\ex\ex\ex\NameauthLName\ex\ex\ex\ex\ex\ex\ex[%
1352     \ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb%
1353     \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1354     \the\@nameauth@etoksd}}%
1355     \ex\ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1356     \ex\ex\ex\ex\ex\ex\ex\ex S\ex\ex\ex\ex\ex\ex\ex\ex\@arga%
1357     \ex\ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex\ex{%
1358     \ex\ex\ex\ex\ex\ex\ex\ex\@nameauth@FirstNametrue%
1359     \ex\ex\ex\ex\ex\ex\ex\ex\NameauthFName\ex\ex\ex\ex\ex\ex\ex[%
1360     \ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb\ex\ex\ex}%
1361     \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1362     \the\@nameauth@etoksd}}%
1363 \fi
1364 \fi
1365 \ignorespaces%
1366 }\ignorespaces%
1367 }\endgroup\ignorespaces}

```

4 Change History

v0.7		\ReverseCommaActive: Added ..	72
General: Initial release	1	\ReverseCommaInactive: Added	72
v0.75		\ReverseInactive: Added	71
\ForgetName: New argument added	86	v1.6	
\IndexName: Current arguments ..	74	nameauth: Environment added ..	90
v0.85		v1.9	
\@nameauth@Name: Comma suppression	65	\ForgetName: Ensure global undef	86
\AKA: Comma suppression	89	\KeepAffix: Added	72
\IndexName: Comma suppression	74	\TagName: Fix cs collisions	82
v0.9		\UntagName: Ensure global undef, fix cs collisions	83
\@nameauth@Suffix: Added	63	nameauth: Bugfix	90
\@nameauth@TrimRoot: Expandable	63	v2.0	
\@nameauth@TrimSuffix: Added	63	\@nameauth@Actual: Added	71
\AKA*: Added	90	\@nameauth@Index: New tagging	70
\FName: Added	74	\@nameauth@Name: Isolate malformed input; trim spaces; redesign tagging	65
\SubvertName: Added	87	\@nameauth@TrimRoot: Trim spaces	63
v0.94		\AKA: Fix malformed input; trim spaces; fix tagging	89
\@nameauth@Hook: Particle caps ..	69	\ExcludeName: Isolate malformed input	78
\@nameauth@Index: Added	70	\ForgetName: Isolate malformed input	86
\CapThis: Added	71	\IndexActual: Added	73
\ExcludeName: Added	78	\IndexName: Isolate malformed input; trim spaces; redesign tagging	74
\IndexActive: Added	73	\PretagName: Added	81
\IndexInactive: Added	72	\SubvertName: Isolate malformed input	87
v0.95		\TagName: Isolate malformed input; redesign tagging	82
\@nameauth@CRii: Added	64	\UntagName: Isolate malformed input; redesign tagging	83
\@nameauth@CapRoot: Added ...	64	General: Use dtxgen template	1
\@nameauth@Hook: Works with microtype	69	nameauth: Redesigned argument handling	90
v1.2		v2.1	
\TagName: Added	82	\@nameauth@CRiii: Added	64
\UntagName: Added	83	\@nameauth@CapRoot: Fix Unicode/NFSS	64
v1.26		\@nameauth@Name: Isolate Unicode issues	65
\@nameauth@CRii: Fixed	64	\AKA: Fix Unicode issues	89
\AKA: Fix name suffixes	89	\AccentCapThis: Added	71
\IndexName: Fix name suffix sorting	74	v2.11	
v1.4		nameauth: Bugfix	90
\@nameauth@Root: More robust ..	63	v2.2	
\ShowComma: Added	72	\NameauthFName: Added	61
v1.5		\NameauthName: Added	61
\@nameauth@AllCapRoot: Added	63		
\@nameauth@Name: Reversing/caps	65		
\@nameauth@TrimSuffix: Trim spaces	63		
\AKA: Reversing and caps	89		
\AllCapsActive: Added	71		
\AllCapsInactive: Added	71		
\CapName: Added	71		
\RevComma: Added	72		
\RevName: Added	71		
\ReverseActive: Added	71		

v2.3		\AKA: Fix index commas, old syntax	89
	\@nameauth@Name: Now internal		65
	\AKA: Expand starred mode		89
	\ExcludeName: Make special xref type		78
	\FName: Interface macro		74
	\FName*: Interface macro		74
	\ForgetName: Global or local		86
	\GlobalNames: Added		72
	\IfAKA: Added		86
	\IfFrontName: Added		85
	\IfMainName: Added		85
	\LocalNames: Added		72
	\Name: Interface macro		74
	\Name*: Interface macro		74
	\NameauthLName: Added		61
	\PName: Work directly with hooks		90
	\SubvertName: Global or local		87
	\ExcludeName: Fix old syntax		78
	\ForgetName: Fix old syntax		86
	\IfAKA: Fix old syntax		86
	\IfFrontName: Fix old syntax		85
	\IfMainName: Fix old syntax		85
	\IndexName: Fix commas, old syntax		74
	\NameAddInfo: Fix old syntax		83
	\NameClearInfo: Fix old syntax		84
	\NameQueryInfo: Fix old syntax		84
	\NoComma: Added		72
	\PretagName: Fix old syntax		81
	\SubvertName: Fix old syntax		87
	\TagName: Fix old syntax		82
	\UntagName: Fix old syntax		83
v2.4		v3.0	
	\@nameauth@Hook: Add hooks	\@nameauth@CRii: Redesigned	64
	\@nameauth@Name: Add token regs for hooks	\@nameauth@CRiii: Redesigned	64
	\AKA: Fix formatting; add token regs	\@nameauth@CapRoot: New UTF test	64
	\FrontNameHook: Added	\@nameauth@Error: Added	64
	\GlobalNames: Ensured to be global	\@nameauth@Hook: Better punctuation detection	69
	\IfAKA: New exclusion test	\@nameauth@Name: Redesigned	65
	\LocalNames: Ensured to be global	\@nameauth@NonWest: Added	67
	\MainNameHook: Added	\@nameauth@Parse: Added	66
	\NameAddInfo: Added	\@nameauth@TagRoot: Added	63
	\NameClearInfo: Added	\@nameauth@TrimRoot: Redesigned	63
	\NameQueryInfo: Added	\@nameauth@TrimSuffix: New test	63
v2.41		\@nameauth@TrimTag: Added	63
	\@nameauth@Name: No local \newtoks	\@nameauth@UTFtest: Added	63
	\AKA: No local \newtoks	\@nameauth@West: Added	68
	nameauth: No local \newtoks	\AKA: Redesigned	89
v2.5		\DropAffix: Added	72
	\@nameauth@Hook: Improve hooks	\ExcludeName: Redesigned	78
	\@nameauth@Name: Hooks query internal values	\ForceFN: Added	72
	\FrontNamesFormat: Added	\IfAKA: Redesigned	86
	General: No default formatting	\IncludeName: Added	79
v2.6		\IncludeName*: Added	80
	\@nameauth@Name: Improve indexing; fix old syntax	\IndexName: Redesigned	74
		\IndexRef: Added	76
		\NameParser: Added	73
		\SeeAlso: Added	73

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	Bess, Good Queen . . .	environments:
<code>\@nameauth@Actual</code> . 436	. . . <i>see</i> Elizabeth I	<code>nameauth</code> . . . 8 , 1285
<code>\@nameauth@AllCapRoot</code>		<code>\ExcludeName</code> . . 26 , 699
. 94	C	
<code>\@nameauth@CRii</code> . . . 120	CAO Cao 50 , 51	F
<code>\@nameauth@CRiii</code> . . 121	<code>\CapName</code> 18 , 440	<code>\FName</code> 15 , 528
<code>\@nameauth@CapRoot</code> 114	<code>\CapThis</code> 19 , 437	<code>\FName*</code> 15 , 529
<code>\@nameauth@CheckDot</code> 134	Carnap, Rudolph . . .	<code>\ForceFN</code> 15 , 446
<code>\@nameauth@Clean</code> . . . 85 23 , 32 , 34	<code>\ForgetName</code> . . 34 , 1121
<code>\@nameauth@Error</code> . . 141	Carter, J.E., Jr., pres.	Francis I, king 55
<code>\@nameauth@EvalDot</code> 136 12 , 29	<code>\FrontNameHook</code> . . 23 , 44
<code>\@nameauth@Hook</code> . . . 354	Carter, Jimmy	<code>\FrontNamesFormat</code> .
<code>\@nameauth@Index</code> . . 402	. . . <i>see</i> Carter, J.E., Jr. 23 , 43
<code>\@nameauth@Name</code> . . . 156	Chaplin, Charlie 42	FUKUYAMA Takeshi . 22
<code>\@nameauth@NonWest</code> 258	CHARLES I, king . . 50 , 51	
<code>\@nameauth@Parse</code> . . 185	Charles the Bald, em-	G
<code>\@nameauth@Root</code> . . . 87	peror . . 14 , 15 , 24	GARBO, Greta 21
<code>\@nameauth@Suffix</code> . . 91	Chiang Kai-shek†, pres. 10	<code>\GlobalNames</code> . . 34 , 460
<code>\@nameauth@TagRoot</code> . 89	Cicero, M.T. . . 14 , 15 , 24	Goethe, J.W. von 57
<code>\@nameauth@TestDot</code> 122	Clemens, Samuel L. . .	Gossett, Louis, Jr. . . . 16
<code>\@nameauth@TrimRoot</code> 88	. . . <i>see</i> Twain, Mark	Grant, Ulysses S., pres. 30
<code>\@nameauth@TrimSuffix</code>	Colfax, Schuyler, v.p. . 30	Gregorio, Enrico 2
. 92	Confucius 14 , 15 , 19 , 24 , 33	Gregory I, pope . . 29 , 36
<code>\@nameauth@TrimTag</code> . 90	Cousot, Patrick 21	Gregory the Great . .
<code>\@nameauth@UTftest</code> . 96	 <i>see</i> Gregory I
<code>\@nameauth@West</code> . . . 306	D	
<code>\@nameauth@toksa</code> . . 47	Dagobert I†, king 10	H
<code>\@nameauth@toksb</code> . . 47	DAVIS, Sammy, JR. . . . 51	Hammerstein, Oskar, II
<code>\@nameauth@toksc</code> . . 47	de la Mare, Walter 19 , 57 16 , 19
	DE' MEDICI, Catherine 20	Harnack, Adolf 57
A	<i>de Smet</i> , Pierre-Jean . . 52	Hearn, Lafcadio 39
<code>\AccentCapThis</code> . 20 , 438	de Soto, Hernando . . 9 , 19	Henry VIII†, king . 10 , 56
ADAMS, John Quincy,	Demetrius I Soter, king 55	Hope, Bob 32 , 36
pres. 50 , 51	<i>Doctor angelicus</i>	Hope, Leslie Townes .
Æthelred II, king 21 , 27 , 56	. . . <i>see</i> Thomas Aquinas <i>see</i> Hope, Bob
ÆDELSTAN, king 50	<i>Doctor mellifluus</i> . <i>see</i>	HOWELL, Thurston,
<code>\AKA</code> 35 , 1239	Bernard of Clairvaux	III* 21
<code>\AKA*</code> 35 , 1281	Dongen, Marc van . . 2 , 65	
<code>\AllCapsActive</code> . 18 , 442	<code>\DropAffix</code> 16 , 455	I
<code>\AllCapsInactive</code> 18 , 441	Du Bois, W.E.B. 41	<code>\if@nameauth@InAKA</code> 47
Andreä, Johann . . 21 , 32	du Cange <i>see</i>	<code>\if@nameauth@InName</code> 47
Arai Akino 18	du Fresne, Charles . . . 39	<code>\IfAKA</code> 33 , 1086
Aristotle 6 , 9	DuBois, W.E.B.	<code>\IfFrontName</code> . 32 , 1063
Arouet, François-Marie	. . . <i>see</i> Du Bois, W.E.B.	<code>\IfMainName</code> . . 32 , 1040
. <i>see</i> Voltaire		<code>\IncludeName</code> . . 27 , 791
Attila the Hun 9 , 27	E	<code>\IncludeName*</code> . . 27 , 826
	Einstein, Albert	<code>\IndexActive</code> . . 25 , 462
B 14 , 15 , 24 , 54	<code>\IndexActual</code> . . 28 , 463
Bernard of Clairvaux . . 40	Elizabeth I, queen 6 , 9 , 37	<code>\IndexInactive</code> . 25 , 461

Iron Mike *see* Tyson, Mike

Ishida Yoko† 18

J

Janos, James
 see Ventura, Jesse

Jean sans Peur, duke . . 36

Jean the Fearless . . .
 see Jean sans Peur

John Eriugena 19

K

Kanno, Yoko† 18

\KeepAffix 16, 456

Kemal, Mustafa 48

Keynes, John Maynard 24

King, Martin Luther,
 Jr. 22

Koizumi Yakumo . . .
 see Hearn, Lafcadio

Konoe, Fumimaro†, PM
 6, 9, 17, 19

Kresge, Joseph *see* Kre-
 skin, The Amazing
Kreskin, The Amazing 39

L

Lao-tzu 36, 40

Leo I, pope 29

Leo the Great . . *see* Leo I
Lewis, Clive Staples .
 5, 9, 15, 54

Li Er *see* Lao-tzu

\LocalNames . . . 34, 459

Louis XIV, king 16, 26, 36

Lueck, Uwe 2, 64

Luecking, Dan 43

Łukasiewicz, Jan 27

M

Maimonides
 38, *see also* Rambam

\MainNameHook . . . 23, 42

Malebranche, Nicolas . 23

Mao Tse-tung†, chair-
 man 19, 56

MENGDE . . *see* CAO Cao

Mill, J.S. 19

Miyazaki Hayao . . 14, 15

Moses ben-Maimon . .
 . . . *see* Maimonides

N

\Name 14, 525

\Name* 14, 526

\NameAddInfo . . 30, 971
nameauth (environ-

ment) 8, 1285

\NameauthFName . . 47, 53

\NameauthLName . . 46, 53

\NameauthName . . 45, 53

\NameClearInfo 31, 1017

\NameParser . . . 49, 466

\NameQueryInfo . 30, 994

\NamesActive . . 23, 458

\NamesFormat . . . 23, 41

\NamesInactive . 23, 457

\NoComma 16, 454

O

Oberdiek, Heiko . . . 2, 62

P

Patton, George S., Jr. . 54

Plato 55

\PName 40, 1282

\PName* 40, 1284

\PretagName . . . 27, 850

Ptolemy I Soter†, king . 56

R

Rambam 38,
 see also Maimonides

\RevComma 19, 447

\ReverseActive . 17, 445

\ReverseCommaActive
 19, 451

\ReverseCommaInactive
 19, 449

\ReverseInactive 17, 444

\RevName 17, 443

Rockefeller, Jay . . .
 . *see* Rockefeller,

 John David, IV
Rockefeller, John David,

 II 5, 9

ROCKEFELLER, John
 David, III . . 50, 51

Rockefeller, John David,
 IV 5, 8, 9

RÜHMANN, Heinrich
 Wilhelm . . . *see*

 RÜHMANN, Heinz

RÜHMANN, Heinz 37

S

Schlicht, Robert . . . 2, 21

\SeeAlso 26, 465

\ShowComma 16, 453

Sine Nomine (Sine Pagina)

Smith, John* 30

Smith, John* (second) . 30

Smith, John* (third) . . 30

Snel van Royen, R. . . . 38

Snel van Royen, W. . . . 38

Snellius *see* Snel
 van Royen, R.;

 Snel van Royen, W.

Stephani, Philipp 2

Strietelmeier, John . . . 18

\SubvertName . . 34, 1180

Sullenberger, Chesley
 B., III 15, 25

Sun King . *see* Louis XIV

Sun Yat-sen, pres. . . 16, 55

T

\TagName 29, 897

Thomas à Kempis 20

Thomas Aquinas 38

Thomas of Aquino . . .
 see Thomas Aquinas

Twain, Mark 40

Tyson, Mike 39

U

\UntagName 30, 948

V

Ventura, Jesse 33

Vlad II Dracul 46

Vlad III Dracula 46

Vlad Țepeș *see*
 Vlad III Dracula

Voltaire 40

W

Washington, George,
 pres. 5, 9, 30, 46, 48

White, E.B. 18, 58

William I 40

William the Conqueror
 *see* William I

Y

Yamamoto Isoroku 6, 9, 17

Yohko 18

Yoshida Shigeru†, PM . 10