

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

Charles P. Schaum[†]

Released 2017/03/22

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 Quick Start	2	2.5.4 Index Sorting	33
1.1 Introduction	2	2.5.5 Index Tags	35
1.2 Basic Concepts	3	2.6 “Text Tags”	36
1.3 Traditional Interface	4	2.7 Name Decisions	37
1.4 Simplified Interface	6	2.7.1 Testing Decisions	37
1.5 Older Syntax	9	2.7.2 Changing Decisions	40
1.6 Reference Tables	10	2.8 Name Variant Macros	42
2 Detailed Usage	13	2.9 Longer Examples	46
2.1 Package Options	13	2.9.1 Variant Names	46
2.2 Naming Macros	16	2.9.2 <code>\LocalNames</code>	48
2.2.1 <code>\Name</code> and <code>\Name*</code>	16	2.9.3 Unicode + <code>inputenc</code>	49
2.2.2 Forenames: <code>\FName</code>	17	2.9.4 \LaTeX Engines	51
2.3 Language Issues	18	2.9.5 Hooks: Intro	52
2.3.1 Affixes Need Commas	18	2.9.6 Hooks: Life Dates	53
2.3.2 Eastern Names	19	2.9.7 Hooks: Advanced	55
2.3.3 Initials	20	2.9.8 Full Redesign	62
2.3.4 Hyphenation	20	2.10 Technical Notes	63
2.3.5 Listing by Surname	21	2.11 Errors and Warnings	64
2.3.6 Particles	21	3 Implementation	65
2.3.7 Accented Names	24	3.1 Flags and Registers	65
2.4 Formatting	24	3.2 Hooks	67
2.4.1 Spaces & Full Stops	24	3.3 Package Options	68
2.4.2 Formatting in the Text	25	3.4 Internal Macros	68
2.4.3 Alternate Format	27	3.5 User Interface Macros	79
2.5 Indexing Macros	30	4 Change History	103
2.5.1 Indexing Control	30	5 Index	105
2.5.2 Index Entries	31		
2.5.3 Index Cross-References	32		

*This file describes version 3.2, last revised 2017/03/22.

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

1 Quick Start

1.1 Introduction

Disclaimer

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

Denotative Signs

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

Design

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

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



Thanks

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

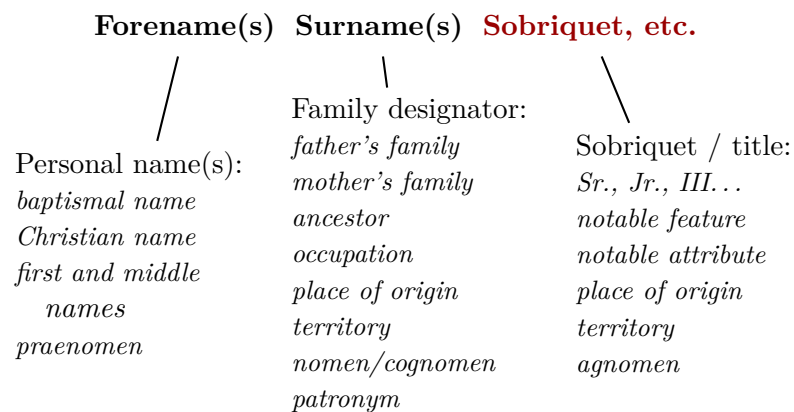
1.2 Basic Concepts

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

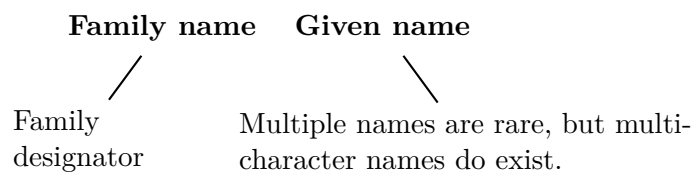
For teaching purposes, we highlight names using sans-serif and use color to show first and subsequent uses of names (see also Sections 2.4.2 and 2.7.2).

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

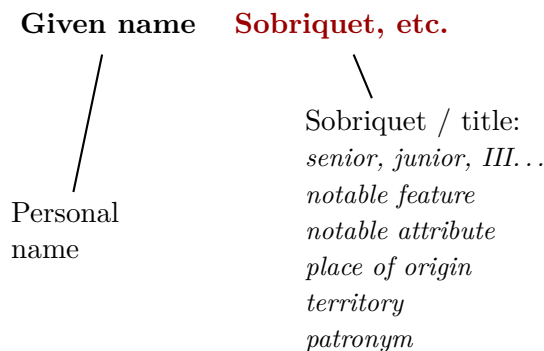
1. Western name: **George Washington**



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



3. Ancient name: **Elizabeth I**



¹Compare [Mulvany, 152–82] and the *Chicago Manual of Style*. That approach is adapted to L^AT_EX and its way of handling optional arguments.

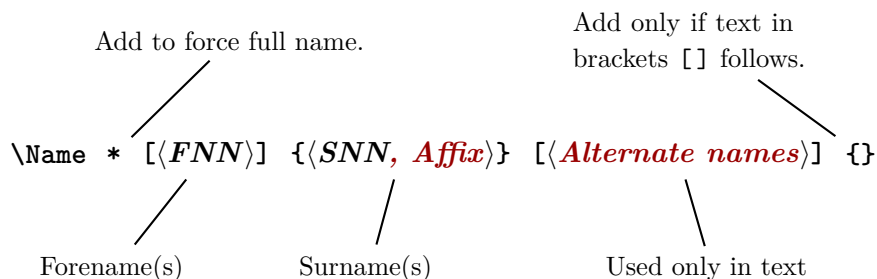
Based on the classes of names, the `nameauth` macros halt with an error in the following cases:

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

1.3 Traditional Interface

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

Western Names



Examples:

One always must include all fields for consistent index entries.

```

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

```

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

```

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

```

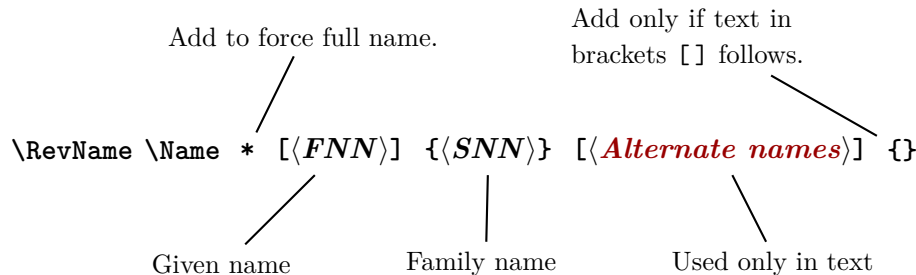
Both affixes and alternate names can vary in the text. Western names require a comma to delimit affixes; see Sections 1.5 and 2.3.1. Using alternate names does not trigger an explicit first use. That is intentional.

```

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

```

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



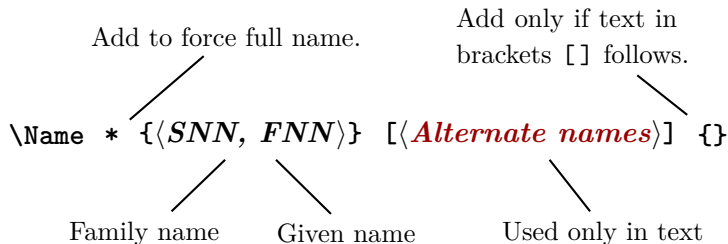
Examples:

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

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

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

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



Examples:

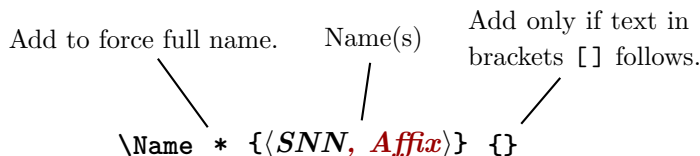
The main feature of non-Western forms in nameauth is the comma-delimited suffix. Eastern names have the family name in $\langle SNN \rangle$ where ancient names have the personal name, but that root name remains the required argument.

These names always take the form $\langle SNN FNN \rangle$ in the index. See Section 2.3.2. In this manual we refer to the “native” Eastern form below:

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

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

Ancient Names



Examples:

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

```

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

```

1.4 Simplified Interface

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

```

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

```

- Field 1 contains text that will be turned into three control sequences. For example, `Wash` generates `\Wash` (like `\Name`): *George Washington*, `\LWash` (L for Long; like `\Name*`): *George Washington*, and `\SWash` (S for Short; like `\FName`): *George*.
- Fields 2 and 3 hold the name arguments.
- Field 4 usually remains empty. It handles the older non-Western syntax (Section 1.5) and permanent alternate names (next page).
- In this context, “\<” is an escape character and a control sequence. If you forget it or just use `<` without the backslash, you will get errors.
- There *must* be four argument fields (three ampersands) per line. Leaving out an ampersand will cause an error.

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

So, why use it?

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

```

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

\IndexName [George]{Washington} . . . . . \JustIndex\Wash

\ForgetName [George]{Washington}%
\Name [George]{Washington} . . . . . \ForgetThis\Wash: George Washington

\SubvertName [George]{Washington}%
\Name [George]{Washington} . . . . . \SubvertThis\Wash: Washington

```

Examples:

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

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

Some Devils in the Details:

English keeps the prefix with the surname in the text and the index, while German keeps particles separate:

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

Martin Van Buren is “Van Buren, Martin” in the index. `\JWG` prints **J.W. von Goethe** and **Goethe**, with “Goethe, J.W. von” in the index. You get a quasi-Anglicized **von Goethe** with `\LJWG[von]`. Either `\CapThis\LJWG[Von]` or `\LJWG[Von]` produce **Von Goethe**; see Section 2.3.6. Additionally, [Mulvany, 152–82] and the *Chicago Manual of Style* offer helpful guidance.

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

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

With the names above you get **Jay Rockefeller IV**, **C.S. Lewis**, **Æthelred Unrædig**, and **Miyazaki Sensei** instead of those from the previous page: **John David Rockefeller IV**, **Clive Staples Lewis**, **Æthelred II**, and **Miyazaki Hayao**.² They all have the same respective index entries and first/subsequent uses, which is why we forced the formatting in the names above. Also `\LLewis[Jack]` prints **Jack Lewis** while `\LCSL[Jack]` prints **C.S. Lewis[Jack]**. Section 2.2.2 explains why.



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



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

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



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

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

1.5 Older Syntax

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

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

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

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

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

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

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

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

1.6 Reference Tables

Getting Things Done

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

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

Form and Format Overview

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

`\Name` or Unmodified Shorthand

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

`\Name*` or L-modifier + Shorthand

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

`\FName` or S-modifier + Shorthand

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

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

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

Selected Macro Patterns:

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

Prefix Macros (One-Time Effect):

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

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

³`\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.

⁴Has no effect on non-Western name forms.

⁵Only affects non-Western name forms.

⁶Only affects Western name forms.

⁷Used best with Western and ancient name forms.

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

2 Detailed Usage

2.1 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. In the listings below, **implicit default options are boldface and need not be invoked by the user.** **Non-default options are in red and must be invoked explicitly.**

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 (see also page 15).

frontmatter Start with “front-matter names” and hooks.

alwaysformat Use only respective “first use” formatting hooks.

formatAKA Format the first use of a name with `\AKA` like the first use of a name with `\Name`.

oldAKA Force `\AKA*` to act like it did before v.3.0.

The `mainmatter` option and the `frontmatter` option enable two different systems of name use and formatting. They are mutually exclusive. `\NamesActive` starts the main matter system when `frontmatter` is used. See Section 2.4.2.

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

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

- 3.0** Using the `oldAKA` option forces `\AKA*` always to print a “forename” field in the text, as it did in versions 2.6 and older. Otherwise the current behavior of `\AKA*` prints in the same fashion as `\FName` (see Sections 2.2.2 and 2.8).

Enable/Disable Indexing

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. **Caution:** using `noindex` and `\IndexInactive` prevents index tags until you call `\IndexActive`, as explained also in Section 2.5.1.



Enable/Disable Index Sorting

pretag	Create sort keys used with <code>makeindex</code> .
nopretag	Do not create sort keys.

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

Affect the Syntax of Names

Show/Hide Affix Commas

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

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

1. The `comma` option globally produces, e.g., **James Earl Carter, Jr.**
2. Section 2.3.1 shows how one can use `\ShowComma` with the `nocomma` option and `\NoComma` with the `comma` option to get per-name results.

Capitalize Entire Surnames

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

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



Still, you can use this package with non-English conventions (just not via these options). You can add, e.g., uppercase or small caps in surnames, formatting them also in the index. See also Sections 2.4.3 and 2.9.7. The simplified interface aids the embedding of control sequences in names. Section 2.3.2 deals with capitalization on a section-level and per-name basis.

Reverse Name Order

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

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

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

Typographic Post-Processing

Formatting Attributes

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

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

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


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

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

Alternate or Continental Formatting

Alternate Syntactic Formatting

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

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

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

¹⁰This package was designed with type hierarchies in mind, although it has become more flexible. See Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60. I drew some inspiration from the typography in Bernhard Lohse, *Luthers Theologie* (Göttingen: Vandenhoeck & Ruprecht, 1995) and the five-volume series by Jaroslav J. Pelikan Jr., *The Christian Tradition: A History of the Development of Doctrine* (Chicago: Chicago UP, 1971–89). Each volume in the series has its own title.

2.2 Naming Macros

Although the formatting hooks do nothing by default, we use them here for teaching purposes. We also force first and subsequent uses as needed. See also Sections 2.4.2 and 2.7.2, which explain the concept in detail.

2.2.1 `\Name` and `\Name*`

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

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

3.0 In the body text, not the index, the `{Alternate names}` field replaces the `{FNN}` field or the `{opt. FNN/Affix}` field if they exist. If neither of the latter exist, then the older non-Western syntax is used (Section 1.5).

```
\begin{nameauth}
  \< Einstein & Albert & Einstein & >
  \< Cicero & M.T. & Cicero & >
  \< Confucius & & Confucius & >
  \< Miyaz & & Miyazaki, Hayao & >
  \< Eliz & & Elizabeth, I & >
\end{nameauth}
```

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

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

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

Back to Section 1.6

2.2.2 Forenames: \FName

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

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

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

<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>\SMiyaz</code>	Miyazaki
<code>\ForceFN\FName{Miyazaki, Hayao}</code> or <code>\ForceFN\SMiyaz</code>	Hayao
<code>\ForceFN\FName{Miyazaki, Hayao}[Sensei]</code> or <code>\ForceFN\SMiyaz[Sensei]</code>	Sensei
<code>\FName{Elizabeth, I}</code> or <code>\SEliz</code>	Elizabeth
<code>\ForceFN\FName{Elizabeth, I}</code>	I
<code>\ForceFN\SEliz[the First]</code>	the First

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

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

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

The drawback lies in remembering that `\Ches` gives us `Chesley B. Sullenberger III`, while `\LSully` produces `Sully Sullenberger III`. Likewise, `\SCSL[Jack]` produces `C.S.[Jack]`. The final field in the `nameauth` environment populates the `⟨Alternate Names⟩` argument, making `[Jack]` normal text.

Back to Section 1.6

¹¹Otherwise you would get poor results with some royal and ancient names.

2.3 Language Issues

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

2.3.1 Affixes Need Commas

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

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



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

`\KeepAffix` In the text only, `\KeepAffix` turns the space between $\langle SNN \rangle$ and $\langle Affix \rangle$ into a non-breaking space. This holds for a Western surname and affix, an ancient name and affix, and a native Eastern family name and personal name.

`\KeepName` In the text only, `\KeepName` turns all spaces between name components $\langle FNN \rangle$, $\langle SNN \rangle$, $\langle Affix \rangle$, and $\langle Alternate\ name(s) \rangle$ into non-breaking spaces. You get no bad breaks with `\KeepName\LJWG[von] von Goethe`.

`\KeepAffix` and `\KeepName` affect all `nameauth` macros that print names in the text. Spaces between multiple names within each name component (think Spanish surnames and French or German forenames) are not affected.

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

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

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

Back to Section 1.6

2.3.2 Eastern Names

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

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

The index entry of this name form looks like $\langle SNN \rangle$, $\langle FNN \rangle$ (including the comma). This type of entry is a Western form. Pick this form also when using Hungarian names. Apologies for needing to enter Hungarian names in reverse, as in `\RevName\Name*[\langle Fre nec \rangle]{\langle Molnár \rangle}` **Molnár Fre nec†**.

native In contrast, there are two general forms of syntax for “native” Eastern name forms, which are indexed as such and appear in Eastern name order in the body text. Apologies for using quasi-Western $\langle SNN \rangle$ and $\langle FNN \rangle$ nomenclature for Eastern names. The new syntax permits alternate names; the old does not:

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

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

`\ReverseActive` In addition to the class options for reversing and capitalization (Section 2.1),
`\ReverseInactive` `\ReverseActive` and `\ReverseInactive` reverse name order for blocks of text and
`\RevName` `\RevName` does that once per name. These macros only affect names in the text. They work also with `\AKA` and its derivatives.

The reverse output mechanism affects full names only. Nevertheless, it does not force full names. Results vary, based on the type of Eastern name forms being used. Non-native forms are shown by a dagger (†):

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

3.0 Creating “last-comma-first” listings by surname (Section 2.3.5) only makes sense with Western names and maybe non-native Eastern names, but not with native Eastern names or ancient forms. That is why native Eastern forms and ancient forms are unaffected by the comma form of reversing.

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

`\AllCapsActive` Using `\AllCapsActive` `\AllCapsInactive` for blocks of text and `\CapName` for specific names, the `nameauth` package allows one to capitalize $\langle SNN \rangle$ in the body text only. These macros also work with `\AKA` and friends. For caps in the text and index see Sections 2.4.3 and 2.9.7.

Below, non-native Eastern forms (first Western, then reversed) are marked with a dagger (\dagger). All other names are in native Eastern, then Western order. Older-syntax forms have a double dagger (\ddagger):

	<code>\CapName</code> only	<code>\CapName\RevName</code>
<code>\Name*[Yoko]{Kanno}</code>	Yoko KANNO \dagger	KANNO Yoko \dagger
<code>\Name*{Arai, Akino}</code>	ARAI Akino	Akino ARAI
<code>\Name*{Ishida}[Yoko]</code>	ISHIDA Yoko \ddagger	Yoko ISHIDA \ddagger
<code>\Name*{Yohko}</code>	YOHKO	YOHKO

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

Back to Section 1.6

2.3.3 Initials

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

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

2.3.4 Hyphenation

In English, some names come from other cultures. These names, like **John Strietelmeier** (`\Name[John]{Strietelmeier}`, index entry skipped) can break badly. One solution consistently uses optional hyphens, while another uses either `babel` or `polyglossia`. If using both solutions with a name, suppress unwanted index entries.

```

\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
% or polyglossia: \newcommand\de[1]{\textgerman{#1}}
\begin{nameauth}
  \< Striet & John & Strie\ -tel\ -meier & >
  \< Strieti & John & \de{Strietelmeier} & >
\end{nameauth}
\PretagName[John]{Strie\ -tel\ -meier}{Strietelmeier, John}
\PretagName[John]{\de{Strietelmeier}}{Strietelmeier, John}

```

In English, some names come from other cultures. These names, like **John Strietelmeier**, (`\Striet`, index entry skipped) could break badly unless handled correctly. In English, some names come from other cultures. These names, like **John Strietelmeier**, (`\Strieti`) could break badly if not handled correctly.

2.3.5 Listing by Surname

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

3.0 These macros do not affect “native” Eastern and ancient name forms. Also, see below how long uses are not always first uses:

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

3.0 Since reversing with commas does not change “native” Eastern and ancient names, we see its effects on “non-native” Eastern names:

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

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

2.3.6 Particles

According to [Mulvany, 165f.] and the *Chicago Manual of Style*, English names with the particles *de*, *de la*, *d'*, *von*, *van*, and *ten* generally keep them with the last name, using varied capitalization. *Le*, *La*, and *L'* always are capitalized unless preceded by *de*. To Anglicize Goethe in the text as *von Goethe*, but indexed under “Goethe, J.W. von,” we use `\LJWG[von]`. `\Name[Catherine de']{Medici}` should be indexed as “Medici, Catherine de’” instead of modern “De Medici.” See also Sections 2.5.1 and especially 2.9.1 for name variants.

non-breaking spaces We recommend inserting `~` or `\nobreakspace` between particles and names to prevent bad breaks.¹² Some particles look very similar. For example, *L'* and *d'* are two separate glyphs each. *L* and *d* are one Unicode glyph each.

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

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

3.2 `\CapThis`, rather, the capitalizing mechanism that it triggers, has undergone a significant overhaul in recent versions of `nameauth`. Earlier versions tried to take a few “shortcuts” that appeared to work. Problems arose with specific cases where capitalization did not work. We have addressed those problems, most of which involved macro expansion.

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

Now, `\CapThis` should work as expected with all of the Unicode characters available in the T1 encoding. Section 2.9.3 has a list, yet see also the table on pages 455–63 in *The Latex Companion*. For a broader set of Unicode characters, consider using `xelatex` and `lualatex`.

Without going into the gory details, it became clear that:

1. There must be one “regular” test for a leading active Unicode character and a separate test when that occurs in a comma-delimited suffix.
2. We cannot use the suffix designed for printing and for testing if we even have a suffix. The test requires a “raw” form of the suffix.
3. The token list test for active Unicode characters can be its own component shared by the two test forms above.
4. One should do one of the two tests, then pick one of two capitalization methods. Keeping everything separate will help the expansion work properly in every case.
5. Every name component is modified. The idea is that you decide to use `\CapThis` in a short name form when the leading element needs to be capitalized. Chances are, you will not need a full name reference with suffix, etc. By capping every element, you have access to caps on demand using any form of short name reference.

`\CapThis` will not cause errors if one uses the `altformat` option and the provided macros for Continental surname formats because that option entirely bypasses the normal in-text capitalization mechanism. `\CapThis` still triggers the alternate capping macros, but the mechanism is different and far more manual. Otherwise `\CapThis` could cause errors in some cases where control sequences in the macro arguments conflict with the capitalization process. See Section 2.4.3.

For another example, we suppose that you want to mention poet e.e. cummings. You might be in a situation where an editor wants: “Cummings’ motif of the goat-footed balloon man has underlying sexual motifs that nevertheless have a childish facade.” We got that form using:

```
\SkipIndex\SubvertThis\CapThis\Name[e.e.]{cummings'}
```

A long-name reference to E.e. Cummings really does not work, nor is it meant to. `\CapThis` is not meant for general situations. Using `\CapName` replaces both the original $\langle SNN \rangle$ and the $\langle SNN \rangle$ created by `\CapThis`. Again, this usage is situation-dependent.

Names are beautiful, yet ambiguous creatures whose forms change greatly, depending on one’s needs and circumstances. This package allows for such variation, yet it provides consistent in the index. We do try to minimize the amount of typing, allowing for automatic reformatting if one moves blocks of text around. We hope that this approach is useful.

`\AccentCapThis`
3.0

If the source files for the `nameauth` package have Unicode encoding and run on a Unicode-compliant system, `\AccentCapThis` is not necessary. See also page 69. If the text encoding of the source files is changed or there are system encoding issues, `\AccentCapThis` might be needed with NFSS when the first name character is an active Unicode character. See also Section 2.9.3.

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

```

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

```

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



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

Using `\CapThis` with forms like `\‘a~Kempis` will work (**À Kempis**) in all situations where one uses the preamble snippet in Section 2.9.4.¹⁵



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

```

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

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

```

Back to Section 1.6

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

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

¹⁵This little example is among one of the longest uses of prefix macros in this manual: `\SkipIndex\CapThis\SubvertThis\ForceFN\ForceName\FName{Thomas, \‘a Kempis}`.

2.4.2 Formatting in the Text

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

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

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

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

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

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

```
\renewcommand*\FrontNamesFormat[1]{\color{magreen}\sffamily #1}
\renewcommand*\FrontNameHook[1]{\color{maolive}\sffamily #1}
\renewcommand*\NamesFormat[1]{\color{mavioiolet}\sffamily #1}
\renewcommand*\MainNameHook[1]{\color{maorange}\sffamily #1}
```

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

We switch to the “front matter” system:

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

Then we switch back to “main matter” system:

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

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

`alwaysformat` Below we simulate the `alwaysformat` option by manipulating the package internals. Using first-use hooks will not force full name references.

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

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

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

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

applied to
footnotes



The previous examples illustrate the independent systems or “species” of names. Use different “species” in different parts of your document. When we do not do this, for example, names in the body text like **John Maynard Keynes** affect names in the footnotes.¹⁶ In this case, `\MainNameHook` is called instead of `\NamesFormat` because the name already occurred in the text.

```
\makeatletter                                % text affects footnotes
\let\@oldfntext\@makefntext                  % restore this later
\long\def\@makefntext#1{%                    % new format; same name system
  \renewcommand*\NamesFormat{\color{naviolet}\scshape}%
  \@oldfntext{#1}}
\makeatother
```

The front-matter system keeps names in the footnotes independent of those in the body text.¹⁷ You can synchronize the two naming systems if needed; see Section 2.7.2. Using the front-matter system looks like:

```
\makeatletter                                % text does not affect footnotes
\long\def\@makefntext#1{%                    % new format; different name system
  \renewcommand*\FrontNamesFormat{\color{nagreen}\scshape}%
  \NamesInactive\@oldfntext{#1}\NamesActive%
}\makeatother
```

Now we change footnotes back to normal, for example:

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

Back to Section 1.6

¹⁶You get **Keynes** from `\Name[John Maynard]{Keynes}` instead of **JOHN MAYNARD KEYNES**.

¹⁷We have the expected **JOHN MAYNARD KEYNES**, then **Keynes**.

2.4.3 Alternate Format

Basic Features

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

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

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

how to break stuff



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

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

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

`\AltFormatActive`

Both the `altformat` option and `\AltFormatActive` globally enable alternate formatting and switch the formatting macros “on.” It will change the effects of `\AltFormatActive*`. It causes `\CapThis` only to work via `\AltCaps`.

`\AltFormatActive*`

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

`\AltFormatInactive`

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

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

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

`\textIT` Three other macros also implement alternate formatting. These macros make

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

`\textUC` demonstrate the formatting activated by `\AltFormatActive`.

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

```

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

```



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

comma karma

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

```

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

```

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

```

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

```

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

Only the new syntax allows one to use alternate names in the text. For example, `\LFukuyama[Sensei]` **FUKUYAMA Sensei** wrote *Nihon Fukuin Rūteru Kyōkai Shi* in 1954, after studying in the US in the 1930s. The old syntax `\LOFukuyama[Sensei]`, which we avoid, yields **FUKUYAMA Takeshi**[Sensei].

Advanced Features

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

```

\Name[Martin]{\textSC{Luther}} basic
\Name[Martin]{\noexpand\textSC{Luther}} advanced

```

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

```

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

```

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

The macros below work together for advanced alternate formatting.

- | | |
|-----------------------|--|
| <code>\AltOff</code> | 1. The macro <code>\AltOff</code> does nothing except when called in a formatting hook, where it “switches off” alternate formatting. When that happens, <code>\textSC</code> , <code>\textBF</code> , <code>\textIT</code> , and <code>\textUC</code> do nothing. This macro works with the <code>altformat</code> option and when <code>\AltFormatActive</code> has been called. |
| <code>\AltOn</code> | 2. The macro <code>\AltOn</code> does nothing except when called in a formatting hook, where it “switches on” alternate formatting. When that happens, <code>\textSC</code> , <code>\textBF</code> , <code>\textIT</code> , and <code>\textUC</code> perform their changes. This macro works when <code>\AltFormatActive*</code> has been called. |
| | 3. Using <code>\noexpand</code> is the golden key (<i>clavis aurea</i>) that lets us expand formatting changes only when desired. It enables this kind of formatting hook, which we must implement: |
| | <code>\renewcommand*\MainNameHook{\AltOff}</code> |
| <code>\AltCaps</code> | 4. Since the normal effects of <code>\CapThis</code> are disabled <code>\AltCaps</code> provides an alternate means to this end. It capitalizes its argument in braces <code>{ }</code> when it is used in a macro hook and triggered by <code>\CapThis</code> . |

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

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

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

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

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

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

Sections [2.4.2](#) and [2.9.7](#) have more on these topics. We resume normal formatting with `\AltFormatInactive`. We do not use alternately-formatted names in the normal regime in order to prevent spurious index entries.

Back to Section [1.6](#)

2.5 Indexing Macros

- 3.0** Current versions of `nameauth` offer greater flexibility with indexing but still implement some error protection. We cover the indexing macros here because the later macros in this manual build on many of their concepts. Some aspects of indexing go beyond the scope of this package.¹⁸

2.5.1 Indexing Control

`\IndexActive` Using the `noindex` option deactivates the indexing function of this package until `\IndexActive` occurs. Another macro, `\IndexInactive`, will deactivate indexing again. These can be used throughout the document. `\ExcludeName` and `\IncludeName` do not deactivate indexing, but they leverage the cross-referencing system to prevent page entries.

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

`\IndexInactive` suppresses index sorting and tagging macros.

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

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



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

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

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

`\Washs` and `\Washes` produce `George Washington's` and `Washington's`, but no index entries. Use `\JustIndex\Wash` as needed. Remember that one only needs this trick when using something other than default formatting. Otherwise just put an inflected ending after the name macro.

Back to Section 1.6

¹⁸For example, search for “memoir babel index” at <http://tex.stackexchange.com>.

2.5.2 Index Entries

`\IndexName` The naming macros (`\Name`, etc.) use this macro to create index entries. You can use it too. It prints nothing in the body text. The syntax is:

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

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

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

The indexing mechanism in the `nameauth` package follows [Mulvany, 152–82] and the *Chicago Manual of Style* regarding Western name affixes. Thus Chesley B. Sullenberger III becomes “Sullenberger, Chesley B., III” in the index.

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

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

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

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

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

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

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

2.5.3 Index Cross-References

`\IndexRef` 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. Section 2.7.2 show how cross-references are independent of other data sets. The syntax is:

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

The name used for the cross-reference is parsed like `\IndexName`. The final argument is neither parsed nor checked to see if a corresponding main entry exists. For example, to cross-reference “Sun King” with **Louis XIV** use: `\IndexRef{Sun King}{Louis XIV}`. To format that reference in the text, use `\AKA` (Section 2.8).

Please see page 45 regarding complex cross-references.

`\SeeAlso` One can precede `\IndexRef`, `\AKA`, or `\PName` with `\SeeAlso` to produce a *see also*
3.0 *also* reference for a name that has appeared already in the index.¹⁹ However, this should be used with caution, as the following points indicate:

- If on page 10 there is `\SeeAlso\IndexRef{Bar}{Foo}`, one *cannot* have index page entries for “Bar” thereafter. A *see also* reference comes after page references.
- If on page 10 there is `\SeeAlso\IndexRef{Bar}{Foo}`, one *can* have index page entries for “Foo” thereafter because it is the target of “Bar.”
- If on page 10 there is `\Name{Bar}` and on page 12 `\IndexRef{Bar}{Foo}`, that will not work because *see* references cannot contain page references.
- Suggestion: Group references together: `\IndexRef{Bar}{Baz; Foo}`. Avoid `\IndexRef{Bar}{Baz} \IndexRef{Bar}{Foo}`.²⁰

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

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

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

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

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

¹⁹When 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.

²⁰Professional indexers often use programs like `Cindex` that enforce a rigorous, standard methodology and syntax. The `nameauth` package likewise tries to follow suit.

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

3.0

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

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

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

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

Back to Section 1.6

2.5.4 Index Sorting

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

Basic Sorting (for Makeindex and More)

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

2.0

```
\PretagName [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 tagged and sorted automatically.

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

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

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

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

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

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

Extra Spaces and Sorting



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

Section 2.9.3 shows how this is related to the number of times the active character must be expanded. The character `Æ` must expand twice, through both `\IeC` and `\T1`, while `ä` expands only once through `\IeC` to a letter. The character `ß` (*scharfes Ess*, *Esszett*) below expands twice.

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

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

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

Back to Section 1.6

2.5.5 Index Tags

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

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

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

<code>\index{Aethelred 2@</code>	<code>Ethelred II</code>	<code>, king}</code>	<code>\TagName and \UntagName</code>
----------------------------------	--------------------------	----------------------	--------------------------------------

All the tagging commands are keyed to the name arguments. `\PretagName` generates the leading sort key while `\TagName` and `\UntagName` affect the trailing content of the index entry.

Tags created by `\TagName` can be helpful in the indexes of history texts, as can other package features. `\TagName` causes the nameauth indexing macros to append “`,_pope`” to the index entries for the popes below:

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

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

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

We see both the old syntax and the new syntax used above. `\TagName` works with all name types, but not with cross-references from `\IndexRef`, etc. Tags are literal text that can be daggers, asterisks, and so on. For example, all fictional names in the index of this manual are tagged with an asterisk. One must add any desired spacing to the start of the tag. Tagging aids scholarly indexing and can include life/regnal dates and other information.



You can use the `{⟨tag⟩}` field of `\TagName` to add specials to index entries for names. Every name in this manual is tagged with at least `|hyperpage` to allow hyperlinks in the index with `ltxdoc` and `hypdoc`. You may have to use `\string|hyperpage` where a vertical bar is active, as in `ltxdoc`.

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

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

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

By using `\TagName` and `\UntagName`, one can disambiguate different people with the same name. For example, using macros from Section 2.7.2:

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

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

The tweaking macros (Section 2.7.2) make it seem like you are dealing with three people who have the same name. The index tags will group together those entries that have the same tag.²¹

Back to Section 1.6

2.6 “Text Tags”

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

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

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

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

For example, `\NameAddInfo[George]{Washington}{(1732--99)}` will associate the text “(1732–99)” with the name “George Washington.” Note, however, that the tag does not print automatically with the name. The tag exists as the value to which a control sequence based on `Washington’s` name expands. Such a tag always must expand in the index to have consistent entries. In the text that is not required, so we do that explicitly with `\NameQueryInfo`.

²¹Since this document, unlike the example above, puts an asterisk by all fictional names in the index, it puts an asterisk at the beginning of the tags above and does not `\UntagName John Smith`, but re-tags him with an asterisk again. We also used `\string|hyperpage` in all the index tags. The information is not shown above for the sake of simplicity and pedagogy.

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

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

Thus, ‘‘`\NameQueryInfo[George]{Washington}`’’ expands to ‘‘(1732–99)’’. As with index tags, one can put a space at the start of a tag—or not. In text tags one might use asterisks, daggers, and even footnotes, such as one for [Schuyler Colfax](#).²² We can include a ‘‘text tag’’ within another one, thus building complex relations. Keeping this in mind, we look at the source for the footnote:

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

Please remember that text tags which query each other or themselves would cause a stack overflow unless you prevented that.

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

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

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

Back to Section [1.6](#)

2.7 Name Decisions

2.7.1 Testing Decisions

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

If one uses these macros inside other macros or passes control sequences to them, the expansion of control sequences can create false results (see *The 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 [2.9.6](#) and [2.9.7](#) for related ideas about tokens and expansion. Using the `trace` package, `\show`, or `\meaning` can help you.

²²Seventeenth vice-president of the US during the first term (1869–73) of [Ulysses S. Grant](#) (president 1869–77).

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

```
\IfMainName[<FNN>]{<SNN>}[<Alternate names>]{<yes>}{<no>}
```

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

For example, we get “I have not met Bob” because we have yet to invoke the name `\Name[Bob]{Hope}`. We will create a manual index entry here.

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

Please note that this test is not affected by the use of `\IndexName`. Since we have encountered [Elizabeth I](#), we get “I met Bess” with a similar example:

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

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

```
\IfFrontName[<FNN>]{<SNN>}[<Alternate names>]{<yes>}{<no>}
```

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

For example, based on [Section 2.4.2](#), we see that “[Carnap](#) is both” a formatted and unformatted name with the following test:

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

Please refer to [Sections 2.7.2](#) and [2.9.2](#) to understand the scope and operation of main- and front-matter names.

This space intentionally left blank.

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

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

This macro works similarly to `\IfMainName`, although it has an additional `⟨excluded⟩` branch in order to detect those names excluded from indexing by `\ExcludeName` (Section 2.5.3).

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

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

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

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

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

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

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

Here we test for a name used with `\ExcludeName` (Section 2.5.3) to get the result, “**Grinch** is excluded”:

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

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

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

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

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

We use `\SkipIndex` to prevent the name “**Freddy's**” from making an index entry of its own. See Section 2.5.1. Take care to avoid a stack overflow by not allowing conditional text to call tags recursively “down the rabbit hole.”

Back to Section 1.6

2.7.2 Changing Decisions

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

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

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

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

	Name Length	Format Hooks	Decision ²²
First Use	Always long	First	False
Subsequent Use	Long or short	Subsequent	True

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

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

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

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

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

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

<code>\SubvertThis\LANth</code>	Susan B. Anthony	<i>force subseq. use, force long</i>
<code>\ForceName\SANth</code>	Susan B.	<i>subseq. use, force first format</i>
<code>\ForgetThis\SANth</code>	Susan B. Anthony	<i>force first use and format</i>
<code>\SANth</code>	Susan B.	<i>subsequent use, short</i>

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

Naming system scope By default, these macros affect a name form in both front matter and main matter naming systems. The example on page 38 above gave us the answer, “Carnap is both.” After we use `\ForgetName[Rudolph]{Carnap}` we get the result: “Rudolph Carnap is not mentioned.” Both front- and main-matter names were forgotten and now we have a first-use situation.

This default “name scope” behavior helps synchronize formatted and unformatted types of names. For example, one could use `\ForgetName` and `\SubvertName` in the footnote examples from Section 2.4.2 to synchronize uses of names between formatting systems. This manual uses that approach at need.

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

Back to Section 1.6

²²Decision outcome prior to the name being used.

2.8 Name Variant Macros

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

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

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

Both macros create a cross-reference in the index from the `<Alt. FNN>`, `<Alt. SNN>`, and `<Alt. names>` fields to a target defined by `<FNN>` and `<SNN>`, regardless of whether that name exists. **The order of the name and cross-reference in `\AKA` is opposite that of `\IndexRef`.**²⁴ See also Section 2.5.5.

`\AKA` only prints whatever form of name in the text that you manually specify. It is designed for the occasional mention of alternate names. See page 45 for alternate solutions. `\SeeAlso` works with `\AKA`, `\AKA*`, and `\PName`.

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

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

General Tips

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

²⁴That ordering is due to the collision between `<Alt1>` and `<FNN2>` in a hypothetical `\AKA[FNN1]{SNN1}[Alt1][FNN2]{SNN2}[Alt2]` By only allowing `<FNN1>` and `<SNN1>` for the target name, we can let the other fields permit an unrestricted cross-reference.

Examples


We make cross-references to [Bob Hope](#), where all of the forms below create the cross-reference “[Hope, Leslie Townes](#) *see* [Hope, Bob](#)”:

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

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

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

Formatting Alternate Names: General

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

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

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

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

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

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

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

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

Formatting Alternate Names: Continental

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

1. Tag the names for proper sorting.

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

2. “Heinz RÜHMANN” is the main name, but we do not start with that. We begin with `\AKA*` in order to use his legal name as an alias for his more popular stage name. `\AKA*` prints “Heinrich Wilhelm” in the body text and sets up the index cross-reference “RÜHMANN, Heinrich Wilhelm *see* RÜHMANN, Heinz.”

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

3. `\SubvertThis` makes `\FName` print “Heinz.”

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

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

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

The resulting text is:

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

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

Advanced Cross-Referencing

- 3.0** `\AKA` will not create multiple cross-references. Handle the special case where one moniker applies to multiple people with `\IndexRef`, e.g., “Snellius” for both [Willebrord Snel van Royen](#) and his son [Rudolph Snel van Royen](#):²⁵

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

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

- Refer to the person intended, e.g.:
`Maimonides` ([Moses ben-Maimon](#)):
`\Name{Maimonides}` (`\AKA{Maimonides}{Moses ben-Maimon}`)
- We now have a main name with a page entry, as well as a “*see* reference” name. If we fail to refer to the main name, we would have a cross-reference to an entry that does not exist.
- Before creating a *see also* cross-reference, one must refer to the alternate name so that all the page entries precede the cross-reference, e.g.:

```
Rambam \Name{Rambam}
```

- 3.0**
- For whatever name you use for the *see also* reference, put the cross-reference after all of the page references. For example, you could put both of these macros at the end of the document:²⁶
`\SeeAlso\IndexRef{Maimonides}{Rambam}`
`\SeeAlso\IndexRef{Rambam}{Maimonides}`
 - You could let the last reference to either name be handled by `\SeeAlso\AKA`, but that could be more confusing and prone to error.

Using `\PretagName` (Section 2.5.4) helps to avoid the need for manual index entries, as the following example shows:

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

Perhaps the greatest medieval theologian was [Thomas Aquinas](#), also known as *Doctor angelicus*.

We use the medieval form: `\Name{Thomas, Aquinas}` because “Aquinas” is not a surname, even though many people, including scholars, use it as such. Section 2.3.6 talks about how one can use `\ForceFN\FName{Thomas, Aquinas}` to refer to [Aquinas](#). Using `\ForceFN\Name{Thomas, Aquinas}` will produce [Thomas](#). That helps prevent unwanted side effects with Eastern names.

²⁵We shorten the index entries via `\Name[W.]{Snel van Royen}[Willebrord]`, and for his son, `\Name[R.]{Snel van Royen}[Rudolph]`.

²⁶Different standards exist for punctuating index entries and cross-references. Check with your publisher, style guide, docs for `xindy` and `makeindex`, and <http://tex.stackexchange.com>.

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

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

Apart from `\SkipIndex`, prefix macros only work on the name given by `⟨FNN⟩` and `⟨SNN⟩`, not on the latter cross-reference. `\SkipIndex` keeps both names out of the index. Below we see the only name types that this macro can handle:

```
\PName[Mark]{Twain}[Samuel L.]{Clemens}      Mark Twain (Samuel L. Clemens)
                                              Twain (Samuel L. Clemens)
\PName*[Mark]{Twain}[Samuel L.]{Clemens}[Sam]  Mark Twain (Sam Clemens)
\PName{Voltaire}[François-Marie]{Arouet}      Voltaire (François-Marie Arouet)
                                              Voltaire (François-Marie Arouet)

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

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

`\PName{William, I}{William, the Conqueror}` gives **William I** (**William the Conqueror**) and **William** (**William the Conqueror**).²⁷ If you use `\PName*`, again you will get the long reference **William I** (**William the Conqueror**).

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

Back to Section 1.6

2.9 Longer Examples

2.9.1 Variant Names

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

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

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

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

²⁷The form `\PName{William, I}{William}[the Conqueror]` works, but we discourage it. Also choose forms like `\PName{Lao-tzu}{Li, Er}` instead of `\PName{Lao-tzu}{Li}[Er]`. Avoiding the older syntax with `\AKA` and `\PName` avoids error.

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

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

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

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

2. This name gives us an extra level of difficulty because the two variants differ only in terms of spaces. They share the same internal representation in the `nameauth` macros: `W.E.B. !DuBois`. We fix this ambiguity by inserting a non-printing control sequence in the alternate form, such as `{Du\empty Bois}`. That prevents “DuBois” from breaking at the end of a line or page. A discretionary hyphen would allow the name to break.²⁸
3. Instead of using `\SkipIndex\AltDuBois` every time we wanted to avoid making an index entry, we create a cross-reference in the index from the alternate name to the canonical name:

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

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

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

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

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

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

With `\ForgetThis\OtherDuBois` we get `W.E.B. DuBois` and `DuBois` thereafter. `\LOtherDuBois` gives us `W.E.B. DuBois`, while with `\SOtherDuBois` we get `W.E.B.` The extra full stop at the end of the sentence was gobbled. We used `\global` to ensure that, regardless of scope, our macros work.

Back to Section 1.6

²⁸Ignoring spaces in names is good because it aids fault tolerance, thereby decreasing spurious index entries. Here we have a special case where this behavior is not useful.

2.9.2 `\LocalNames`

As mentioned previously in Section 2.7.2, both `\ForgetName` and `\SubvertName` usually affect both main-matter and front-matter names. This default behavior can be quite helpful. Nevertheless, there are cases where it is undesirable. This section shows `\Localnames` and `\Globalnames` in action, limiting the behavior of the “tweaking macros” to either the main or front matter.

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

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

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

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

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

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

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

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

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

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

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

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

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

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


2.9.3 Unicode + inputenc

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

À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß
à á â ã ä å æ	ç è é ê ë	ì í î ï ð ñ
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ
Ă ă Ą ą Ć ć Č č	Ď ě Đ đ Ę ę Ě ě	Ĝ ĝ Ĩ ĩ
Ĭ ĵ Ĺ ĺ	Ń ń Ņ ņ Œ œ	Ř ř Ŕ ŕ
Ś ś Ŝ ŝ Ţ ţ Ţ ţ	Ũ ũ Ú ú	Ž ž Ž ž Ž ž

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

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

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

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

Argument	Macro	Result
abc	<code>\foo abc\relax</code>	<code><ab><c></code>
{æ}bc	<code>\foo {æ}bc\relax</code>	<code><æb><c></code>
\ae bc	<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
æbc	<code>\foo æbc\relax</code>	xelatex	<code><æb><c></code>
æbc	<code>\foo æbc\relax</code>	lualatex	<code><æb><c></code>
æbc	<code>\foo æbc\relax</code>	pdflatex	<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 69 shows the test. Essentially, the following two expressions are equal under NFSS:

```
\car<test_1>\@nil, where <test_1> expands to \IeC {<test_1>}
\car<test_2>\@nil, where <test_2> expands to \IeC {<test_2>}
```

If `<test_2>` expands to the letter `<test_2>`, then it will fail the test for equality. “Active” characters expand to “two-argument wide” values under NFSS, as the table below shows via defining a macro to be a character, then printing its `\meaning` in the cell:

<code>\def\A{<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.

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

`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 2.3.6 and 2.3.7, as well as Section 2.5.4.

2.9.4 L^AT_EX Engines



The nameauth package tries to work with multiple languages and typesetting engines. The following preamble snippet illustrates how that can be done:²⁹ This example reflects changes to several packages since 2014 and may not address older documents and systems or all possible cases. Of course, the user must specify the main and alternate languages and any package options as the respective package documentation files indicate.

```
\ifdefined\Umathchar
  \usepackage{fontspec}
  \usepackage{polyglossia}
\else
  \usepackage[utf8]{inputenc}
  \usepackage[TS1,T1]{fontenc}
  \usepackage{babel}
\fi
% Below is optional; use only if your dvi viewer
% crashes or becomes unresponsive with tikz.
\usepackage{ifxetex}
\usepackage{ifluatex}
\usepackage{ifpdf}
\ifxetex
  \usepackage{tikz}
\else
  \ifpdf
    \usepackage{tikz}
  \fi
\fi
```

This general arrangement works 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. One must include the `ifxetex`, `ifluatex`, and `ifpdf` packages for it to work.

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

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

2.9.5 Hooks: Intro

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

Margin
Paragraphs



Before we get to the use of text tags and name conditionals in name formatting, we begin with an intermediate example to illustrate that something more complex can occur in `\NamesFormat`. Here we put the first mention of a name in boldface, along with a marginal notation if possible:³⁰

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

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

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

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

Vlad III Dracula
Vlad II Dracul

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

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



You cannot re-enter `\Name` or `\AKA` by calling them within any of the formatting hooks, as the next example shows:

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

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

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

2.9.6 Hooks: Life Dates

We can use name conditionals (Section 2.7.1) and text tags (Section 2.6) to add life information to names when desired.

`\if@nameauth@InName` The example `\NamesFormat` below adds a text tag to the first occurrences
`\if@nameauth@InAKA` 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
`\@nameauth@toksb` conditional macros as we do below. Using these registers allows accented names
`\@nameauth@toksc` to be recognized properly. In `\AKA` the token registers are copies of the *last* three arguments, corresponding to the pseudonym. Nevertheless, they have the same names as the registers in `\@nameauth@Name` because they work the same way and may be easier to use this way.



We assume that we will not be using the `alwaysformat` option, meaning that we only call this hook once for a first use of `\AKA`. We also use a different formatting for the naming macros on the one hand and `\AKA` on the other:³¹

```

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

```

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

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

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

```

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

```

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

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

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

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

```
\ForgetThis\Wash held office 1789--97. No tags: \Wash.
First use, dates suppressed: \NoTagtrue\ForgetThis\Wash.
George Washington (1732–99) held office 1789–97. No tags: Washington.
First use, dates suppressed: George Washington.
```

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

```
\Name[Mustafa]{Kemal} was granted the name%
\begingroup\let\MainNameHook\NamesFormat%
\AKA[Mustafa]{Kemal}{Atatürk}\endgroup. We mention%
\AKA[Mustafa]{Kemal}{Atatürk} again.
Mustafa Kemal (1881–1938) was granted the name Atatürk (in 1934, a
special surname). We mention Atatürk again.
```



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

```
\ForgetName[Mustafa]{Kemal}% first use
\makeatletter\@nameauth@AKAFormattrue\makeatother% formatAKA
\Name[Mustafa]{Kemal} was granted the name%
\AKA[Mustafa]{Kemal}{Atatürk}. We mention%
\AKA[Mustafa]{Kemal}{Atatürk} again.
Mustafa Kemal (1881–1938) was granted the name Atatürk (in 1934, a
special surname). We mention Atatürk again.
```

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

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

See Section 3.4 and page 82 for the decision paths and the logic used by the package. Presently, writing hook macros should be much simpler than in earlier versions of this package.

Back to Section 1.6

2.9.7 Hooks: Advanced

Alternate Formatting

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

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

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

Continental Format

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

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

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

Using this method, formatting occurs in both the text and in the index if the `altformat` option or `\AltFormatActive` was used. If you use a name that uses these macros both within and outside of the alternate formatting regime, you will get spurious index entries.³²

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

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

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

³²Using `\AltFormatActive*` is interesting because it looks like the normal `nameauth` regime but prevents `\CapThis` from having its normal effect unless you use `\AltCaps`. With `\AltFormatActive*` if you use a name that has alternate formatting both within and outside of the alternate formatting regime, you may not get spurious index entries as long as control sequences are consistent.

```

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

```

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

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

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

```

\renewcommand*\MainNameHook{\AltOff}

```

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

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

```

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

```

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

```

\PretagName[John]{\noexpand\textSC{Adams}}{Adams, John}
\PretagName[Sammy]%
  {\noexpand\textSC{Davis}, \noexpand\textSC{Jr}.}%
  {Davis, Sammy, Jr.}
\PretagName{Harun, \noexpand\textSC%
  {\noexpand\AltCaps{a}l-Rashid}}{Harun al-Rashid}
\PretagName{\noexpand\textSC{Mencius}}{Mencius}

```

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

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

- Punctuation detection works: Sammy DAVIS JR. Also Sammy Davis Jr. Then DAVIS. Now Davis. (We used `\ForceName` for formatting.)
- `\ForceName\DropAffix\LSDJR` gives Sammy DAVIS. Otherwise, using just `\DropAffix\LSDJR` gives Sammy Davis.
- `\RevComma\LAdams` yields Adams, John. All the reversing macros work.
- `\ForceName\ForceFN\SHAR` produces AL-RASHID. `\ForceFN\SHAR` produces al-Rashid. If we add `\CapThis` we get AL-RASHID and Al-Rashid. The way that Continental resources treat certain affixes relates to similar issues in [Mulvany, 168–73].³³
- One must include the extra control sequences in all the macro arguments that use these names.

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

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

Rolling Your Own: New Style

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



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

We recommend using the internal package flag `\@nameauth@DoAlt`, which activates alternate formatting, `\@nameauth@DoCaps`, which handles capitalization, and `\@nameauth@InHook`, which is true when the formatting hooks are called. See page 80 and following. If you create your own macros, they will look similar.³⁴

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

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

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

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

³³Handling non-Western names in Western sources can be a gray area. One ought take care to be culturally sensitive in these matters.

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

```

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

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

```

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

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

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

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

Rolling Your Own: Old Style

“Old style” refers to the way hooks were designed before recent package changes. Sometimes one might want to achieve more customized results. We begin that journey by looking at `\NameParser`.

`\NameParser`
3.1

This user-accessible parser (page 82) builds a name from the internal macros `\FNN`, `\SNN`, `\rootb` and `\suffb`. Reversing and commas are still usable; capitalization depends on the context. The general form is:

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

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

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

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

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

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



We have shown already that you do not really need `\NameParser` to use these switching macros in the hooks. Yet the user-level parser does have some handy uses, especially as we go further toward designing custom macros. For example,

we demonstrate an extreme case based on Section 2.9.5 where we modify some internal flags to have `\NameParser` to produce different syntactic forms than the normal output.³⁵

```

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

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

```

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

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

```

\if@nameauth@FullName          Print a full name if true.
\if@nameauth@FirstName        Print a first name if true.
Only one or the other of these can be true to avoid undocumented behavior.

\if@nameauth@RevThis          Reverse name order if true.
\if@nameauth@EastFN          toggled by \ForceFN.
\if@nameauth@RevThisComma    Reverse Western name, add comma.
Reversing without commas overrides reversing with commas.

```



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

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

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

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

We define three Boolean flags and set one of them true by default. The `\ifFbox` flag takes over the internal function of `\@nameauth@DoAlt`, which is enabled by `\AltFormatActive`. The `\ifFirstCap` flag takes over the internal function of `\@nameauth@DoCaps`, which is enabled by `\CapThis`. The `\ifInHook` flag replaces the internal function of `\@nameauth@InHook`, which is enabled by the internal format hook dispatcher.³⁶

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

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

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

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

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

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

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

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

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

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

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

³⁶A similar version of this example is in `examples.tex`, collocated with this manual.

We avoid spurious index entries in the front matter by using the same hooks.

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

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

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

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



We can reuse new-style names with old-style macros when needed. We show this here in abbreviated fashion. We keep the Boolean flags `\ifFirstCap` and `\ifInHook` from earlier. We also keep the redefined `\AltCaps`, `\CapThis`, and `\NamesFormat`. One might have to make modifications as needed.³⁷

```
\newif\ifCaps
\Capstrue
\renewcommand*\textSC[1]{%
  \ifCaps\textsc{#1}\else#1\fi
}
\renewcommand*\MainNameHook[1]
{%
  \Capsfalse\InHooktrue\NameParser\InHookfalse%
  \global\FirstCapfalse\Capstrue%
}
\let\FrontNameHook\MainNameHook
```

The names below have the same declarations and index entries as they did above. They look and work the same but use different macros.

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

As earlier, punctuation detection works: Sammy DAVIS JR. Also Sammy Davis Jr. Then DAVIS. Now Davis. `\ForceName\DropAffix\LSDJR` gives Sammy DAVIS. `\DropAffix\LSDJR` gives Sammy Davis. `\RevComma\LAdams` yields Adams, John. `\ForceName\ForceFN\SHAR` produces AL-RASHID. `\ForceFN\SHAR` produces al-Rashid. If we add `\CapThis` we get AL-RASHID and Al-Rashid.

Use names with alternate formatting only when it is active to avoid spurious index entries. We resume normal formatting with `\AltFormatInactive`.

Back to Section [1.6](#)

³⁷A fuller version of this example is in `examples.tex`, collocated with this manual.

2.9.8 Full Redesign



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

`\NameauthName` These macros are set by default to `\@nameauth@Name`, the internal name parser.
`\NameauthLName` The main and simplified interfaces call them as respective synonyms for `\Name`,
`\NameauthFName` `\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}
  \langle Silly & No Particular & Name & \rangle
\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`.

This space intentionally left blank.

2.10 Technical Notes

About the package itself:

- We put great weight on being backward-compatible with older versions.
 - Recent changes aim for simpler work flow, not more features.
 - The package works with both `xindy` and `makeindex`. We recommend `xindy` for languages whose collating sequences do not map to English.³⁸
- 3.0**
- We support alternate names in both Western and “native” Eastern forms. Mononyms and the older syntax for non-Western names do not support alternate names.
- 3.0**
- Name output, index entries, and index cross-references are independent modules.
- 3.0**
- Warnings for the indexing macros are suppressed unless one uses the `verbose` option. The `nameauth` environment will continue to emit warnings as needed.
- 2.6**
- The `comma` option and the older syntax are no longer restrictive, save with `\AKA` and its derivatives. See Sections 1.5, 2.3.1, and 2.8.
- 2.5**
- No formatting is selected by default. Cf. Sections 2.4.2, 2.9.5, 2.9.6, and 2.9.7.

About the manual:

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

About package building:

- The `nameauth` package requires `etoolbox`, `suffix`, `trimspaces`, and `xargs`. The `dtx` file encoding is UTF-8; we cannot guarantee building and using this package on systems that are not Unicode-compliant.
- With each release, we test `nameauth` with dvi-mode `latex` and with pdf-mode engines `pdflatex`, `lualatex`, and `xelatex` using `makeindex`. We run the GNU Makefile with the `ENGINE=engine` option.³⁹
- This package was built with `pdflatex`. This item changes per \LaTeX engine.
- This package is tested on Ubuntu Linux and Windows 7 (both vanilla \TeX Live). Cygwin provides `make` on Windows. The `pdflatex` version of this package is released from the Ubuntu platform to CTAN.

³⁸`\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.

³⁹The manual is used as the test suite. In dvi mode the manual omits all references to *TikZ* because some dvi display programs (*e.g.* `dviout`, but not `xdvi`) will emit errors about bad specials even if one just includes the `tikz` package. The *TikZ* diagrams herein will appear as blank space in that case. This does not affect `nameauth` proper.

2.11 Errors and Warnings

Here are some ways to avoid common errors:

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

The older syntax presents its own group of potential errors:

- Erroneously typing `\Name[Henry]{VIII}` prints “Henry VIII” and “VIII,” as well as producing a malformed index entry.
- Avoid forms like `\Name[Henry]{VIII}[Tudor]` which gives “Tudor VIII” and “VIII.” That is a Western alternate name form, which is incorrect.
- The older syntax will not work with some macros. The comma-suffixed form does work with those macros. See Section 2.8.

Warnings result from the following:

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

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

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

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

3 Implementation

3.1 Flags and Registers

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

Who Called Me?

These values are used by the format hook dispatcher `\@nameauth@Hook` and the hook macros to determine if they have been called by either `\@nameauth@Name`, `\AKA`, or `\IndexRef`, respectively. Those macros set these flags. On their use, see also Sections 2.9.6 and 2.9.7.

```
1 \newif\if@nameauth@InAKA
2 \newif\if@nameauth@InName
3 \newif\if@nameauth@Xref
```

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

Core Macro Lock

The macros `\@nameauth@Name` and `\AKA`, with some auxiliary macros, process names in a “locked” state. These flags prevent a stack overflow. See also Sections 2.9.6 and 2.9.7.

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

Indexing

As the naming macros have locks, so do the indexing macros. These locks permit or prevent both indexing and tags. `\IndexActive` and `\IndexInactive` or the `index` and `noindex` options toggle the first flag; `\SkipIndex` toggles the second. `\JustIndex` toggles the third, which makes the core naming engine act like a call to `\IndexName`:

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

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

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

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

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

Formatting

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

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

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

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

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

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

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

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

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

Affix Commas

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

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

Name Breaking

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

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

Detect Punctuation

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

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

Long and Short Names

`\if@nameauth@FullName` is true for a long name reference. `\if@nameauth@FirstName` disables full-name references and causes only Western forenames to be displayed.

`\if@nameauth@AltAKA` is toggled respectively by `\AKA` and `\AKA*` to print a longer or shorter name. `\if@nameauth@OldAKA` forces the pre-3.0 behavior of `\AKA*`.

`\if@nameauth@ShortSNN` is used with `\DropAffix` to suppress the affix of a Western name. `\if@nameauth@EastFN` toggles the forced printing of Eastern forenames.

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

Eastern Names

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

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

Last-Comma-First

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

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

Capitalize First Letter

The next flags deal with first-letter capitalization. The first Boolean value is triggered by `\CapThis` and reset by `\Name` and `\AKA`. The second is triggered by `\@nameauth@UTFtest` when it encounters a Unicode character under NFSS. The third is an “override switch” triggered by `\AccentCapThis` as a fall-back. The fourth prevents the first-letter capping mechanism from interacting with Continental formatting and the fifth toggles it.

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

```

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

```

Warning Levels

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

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

Name Argument Token Registers

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

```

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

```

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

```

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

```

3.2 Hooks

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

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

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

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

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

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

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

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

`\NameauthName` Hook to create custom naming macros. Usually the three macros below have the same control sequence, but they need not do so if you want something different. See Section [2.9.8](#). Use at your own risk! Changing these macros basically rewrites this package.

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

`\NameauthLName` Customization hook called after `\@nameauth@FullName` is set true. See Section [2.9.8](#).

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section [2.9.8](#).

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

3.3 Package Options

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

```
55 \DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
56 \DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
57 \DeclareOption{mainmatter}{\@nameauth@MainFormatrue}
58 \DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
59 \DeclareOption{formatAKA}{\@nameauth@AKAFormatrue}
60 \DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
61 \DeclareOption{index}{\@nameauth@DoIndextrue}
62 \DeclareOption{noindex}{\@nameauth@DoIndexfalse}
63 \DeclareOption{pretag}{\@nameauth@Pretagtrue}
64 \DeclareOption{nopretag}{\@nameauth@Pretagfalse}
65 \DeclareOption{allcaps}{\@nameauth@AllCapstrue}
66 \DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
67 \DeclareOption{allreversed}%
68   {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
69 \DeclareOption{allrevcomma}%
70   {\@nameauth@RevAllfalse\@nameauth@RevAllCommatrue}
71 \DeclareOption{notreversed}%
72   {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
73 \DeclareOption{alwaysformat}{\@nameauth@AlwaysFormatrue}
74 \DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
75 \DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
76 \DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
77 \DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
78 \DeclareOption{verbose}{\@nameauth@Verbosetrue}
79 \DeclareOption{altformat}{%
80   \@nameauth@AltFormatrue\@nameauth@DoAlttrue}
81 \ExecuteOptions%
82   {nocomma,mainmatter,index,pretag,%
83     normalcaps,notreversed,noformat}
84 \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.

```
85 \RequirePackage{etoolbox}
86 \RequirePackage{suffix}
87 \RequirePackage{trimspaces}
88 \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.

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

Parsing: Root and Suffix

<code>\@nameauth@Root</code>	The following two macros return everything before a comma in $\langle SNN \rangle$. 91 <code>\newcommand*\@nameauth@Root[1]{\@nameauth@@Root#1,\}</code>
<code>\@nameauth@@Root</code>	Throw out the comma and suffix, return the radix. 92 <code>\def\@nameauth@@Root#1,#2\{\trim@spaces{#1}}</code>
<code>\@nameauth@TrimTag</code>	The following two macros return everything before a vertical bar () in an index tag. 93 <code>\newcommand*\@nameauth@TrimTag[1]{\@nameauth@@TrimTag#1 }</code>
<code>\@nameauth@@TrimTag</code>	Throw out the bar and suffix, return the radix. 94 <code>\def\@nameauth@@TrimTag#1 #2\{\#1}</code>
<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. 95 <code>\newcommand*\@nameauth@Suffix[1]{\@nameauth@@Suffix#1,,}</code>
<code>\@nameauth@@Suffix</code>	Throw out the radix; return the suffix with no leading spaces. We use this when printing the suffix. 96 <code>\def\@nameauth@@Suffix#1,#2,#3\%</code> 97 <code>{\ifx\#2\@empty\else\trim@spaces{#2}\fi}</code>
<code>\@nameauth@GetSuff</code>	The following two macros just grab the suffix for testing if the first non-space character is an active character from <code>inputenc</code> . 98 <code>\newcommand*\@nameauth@GetSuff[1]{\@nameauth@@GetSuff#1,,}</code>
<code>\@nameauth@@GetSuff</code>	Throw out the radix; return the suffix. 99 <code>\def\@nameauth@@GetSuff#1,#2,#3\{\#2}</code>

Parsing: Capitalization

<code>\@nameauth@TestToks</code>	Test if the leading token is the same as the leading token of an active Unicode character, using an <i>Esszett</i> (β) as the control. We only run this macro if we are in the <code>inputenc</code> regime. 100 <code>\newcommand*\@nameauth@TestToks[1]</code> 101 <code>{%</code> 102 <code>\toks@\expandafter{\@car#1\@nil}%</code> 103 <code>\edef\one{\the\toks@}%</code> 104 <code>\toks@\expandafter{\@car\beta\@nil}%</code> 105 <code>\edef\two{\the\toks@}%</code> 106 <code>\ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi</code> 107 <code>}</code>
<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 <code>inputenc</code> is loaded. We set up the comparison and pass off to <code>\nameauth@TestToks</code> . 108 <code>\newcommand*\@nameauth@UTFtest[1]</code> 109 <code>{%</code> 110 <code>\def\testarg{#1}%</code> 111 <code>\ifdefined\Umathchar</code> 112 <code>\@nameauth@UTFfalse%</code> 113 <code>\else</code> 114 <code>\ifdefined\UTFviii@two@octets</code> 115 <code>\if@nameauth@Accent</code>

```

116         \@nameauth@UTFtrue\@nameauth@Accentfalse%
117     \else
118         \expandafter\@nameauth@TestToks\expandafter{\testarg}%
119     \fi
120 \else
121     \@nameauth@UTFfalse%
122 \fi
123 \fi
124 }

```

`\@nameauth@UTFtestS` This test is like the one above, but a special case when we have a suffix. We have to do a bit more in the way of expansion to get the comparison to work properly. Moreover, we only use this when the regular suffix macro is not `\@empty`.

```

125 \newcommand*\@nameauth@UTFtestS[1]
126 {%
127     \let\ex\expandafter%
128     \ex\def\ex\testarg\ex{\@nameauth@GetSuff{#1}}%
129     \ex\toks@\ex\ex\ex{\testarg}%
130     \ex\def\ex\test@rg\ex{\the\toks@}%
131     \ifdefined\Umathchar
132         \@nameauth@UTFfalse%
133     \else
134         \ifdefined\UTFviii@two@octets
135             \ifnameauth@Accent
136                 \@nameauth@UTFtrue\@nameauth@Accentfalse%
137             \else
138                 \expandafter\@nameauth@TestToks\expandafter{\test@rg}%
139             \fi
140         \else
141             \@nameauth@UTFfalse%
142         \fi
143     \fi
144 }

```

`\@nameauth@Cap` The following two macros cap the first letter of the argument.

```

145 \newcommand*\@nameauth@Cap[1]{\@nameauth@C@p#1\}

```

`\@nameauth@C@p` Helper macro for the one above.

```

146 \def\@nameauth@C@p#1#2\%
147   {\expandafter\trim@spaces\expandafter{\MakeUppercase{#1}#2}}

```

`\@nameauth@CapUTF` The following two macros cap the first active Unicode letter under inputenc.

```

148 \newcommand*\@nameauth@CapUTF[1]{\@nameauth@C@pUTF#1\}

```

`\@nameauth@C@pUTF` Helper macro for the one above.

```

149 \def\@nameauth@C@pUTF#1#2#3\%
150   {\expandafter\trim@spaces\expandafter{\MakeUppercase{#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.

```
151 \newcommand*\@nameauth@TestDot[1]
152 {%
153   \def\TestDot##1.\TestEnd##2\{\TestPunct{##2}}%
154   \def\TestPunct##1{%
155     \ifx\TestPunct##1\TestPunct%
156     \else
157       \@nameauth@Puncttrue%
158     \fi
159   }%
160   \@nameauth@Punctfalse%
161   \TestDot#1\TestEnd.\TestEnd\}%
162 }
```

`\@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.

```
163 \newcommand*\@nameauth@CheckDot%
164   {\futurelet\@token\@nameauth@EvalDot}
```

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

```
165 \newcommand*\@nameauth@EvalDot%
166 {%
167   \let\@period=.%
168   \ifx\@token\@period\expandafter\@gobble \fi
169 }
```

Error Detection

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

```
170 \newcommand*\@nameauth@Error[2]
171 {%
172   \edef\msgga{#2 SNN field empty}%
173   \edef\msggb{#2 SNN field malformed}%
174   \protected@edef\testname{\trim@spaces{#1}}%
175   \protected@edef\testroot{\@nameauth@Root{#1}}%
176   \ifx\testname\@empty
177     \PackageError{nameauth}{\msgga}%
178   \fi
179   \ifx\testroot\@empty
180     \PackageError{nameauth}{\msggb}%
181   \fi
182 }
```

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.

```
183 \newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
184 {%
```

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

```
185 \unless\if@nameauth@Lock
186 \@nameauth@Locktrue%
```

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

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

If we use \JustIndex then skip everything else..

```
189 \if@nameauth@JustIndex
190 \IndexName[#1]{#2}[#3]%
191 \@nameauth@InNamefalse%
192 \@nameauth@Lockfalse%
193 \@nameauth@JustIndexfalse%
194 \else
```

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

```
195 \if@nameauth@Forget
196 \ForgetName[#1]{#2}[#3]%
197 \else
198 \if@nameauth@Subvert
199 \SubvertName[#1]{#2}[#3]%
200 \fi
201 \fi
202 \leavevmode\hbox{}%
203 \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
204 \if@nameauth@MainFormat
205 \@nameauth@Parse[#1]{#2}[#3]{!MN}%
206 \else
207 \@nameauth@Parse[#1]{#2}[#3]{!NF}%
208 \fi
209 \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
```

Reset all the “per name” Boolean values.

```
210 \@nameauth@SkipIndexfalse%
211 \@nameauth@Forgetfalse%
212 \@nameauth@Subvertfalse%
213 \@nameauth@Lockfalse%
214 \@nameauth@InNamefalse%
215 \@nameauth@NBSPfalse%
216 \@nameauth@NBSPXfalse%
217 \@nameauth@DoCapsfalse%
218 \@nameauth@Accentfalse%
219 \@nameauth@AllThisfalse%
220 \@nameauth@ShowCommafalse%
221 \@nameauth@NoCommafalse%
222 \@nameauth@RevThisfalse%
223 \@nameauth@RevThisCommafalse%
224 \@nameauth@ShortSNNfalse%
225 \@nameauth@EastFNfalse%
226 \fi
```

Close the “locked” branch.

```
227 \fi
```


Call the full stop detection.

```
228 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
229 }
```

`\@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.

```
230 \newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
231 {%
232 \if@nameauth@Lock
233 \let\ex\expandafter%
```

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

```
234 \protected@edef\arga{\trim@spaces{#1}}%
235 \protected@edef\rootb{\@nameauth@Root{#2}}%
236 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
237 \protected@edef\argc{\trim@spaces{#3}}%
```

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

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

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

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

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

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

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

```
247 \if@nameauth@DoCaps
248 \let\carga\arga%
249 \let\crootb\rootb%
250 \let\csuffb\suffb%
251 \let\cargc\argc%
252 \unless\if@nameauth@AltFormat
```

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

```
253 \unless\ifx\arga\@empty
254 \def\test{#1}%
255 \ex\@nameauth@UTFtest\ex{\test}%
256 \if@nameauth@UTF
257 \ex\def\ex\carga\ex{\ex\@nameauth@CapUTF\ex{\test}}%
258 \else
259 \ex\def\ex\carga\ex{\ex\@nameauth@Cap\ex{\test}}%
260 \fi
261 \fi
```

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

```
262 \def\test{#2}%
263 \ex\@nameauth@UTFtest\ex{\test}%
264 \if@nameauth@UTF
265 \ex\def\ex\crootb\ex{\ex\@nameauth@CapUTF\ex{\rootb}}%
266 \else
```

```

267         \ex\def\ex\crootb\ex{\ex\@nameauth@Cap\ex{\rootb}}%
268         \fi

```

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

```

269         \unless\ifx\suffb\@empty
270         \def\test{#2}%
271         \ex\@nameauth@UTFtestS\ex{\test}%
272         \protected@edef\test{\@nameauth@GetSuff{#2}}%
273         \if@nameauth@UTF
274         \protected@edef\test{\@nameauth@Suffix{#2}}%
275         \ex\def\ex\csuffb\ex{\ex\@nameauth@CapUTF\ex{\test}}%
276         \else
277         \edef\test{\@nameauth@Suffix{#2}}%
278         \ex\def\ex\csuffb\ex{\ex\@nameauth@Cap\ex{\test}}%
279         \fi
280         \fi

```

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

```

281         \unless\ifx\argc\@empty
282         \def\test{#3}%
283         \ex\@nameauth@UTFtest\ex{\test}%
284         \if@nameauth@UTF
285         \ex\def\ex\cargc\ex{\ex\@nameauth@CapUTF\ex{\test}}%
286         \else
287         \ex\def\ex\cargc\ex{\ex\@nameauth@Cap\ex{\test}}%
288         \fi
289         \fi
290         \fi
291         \let\arga\carga%
292         \let\rootb\crootb%
293         \let\suffb\csuffb%
294         \let\argc\cargc%
295         \fi

```

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

```

296         \if@nameauth@AllThis
297         \protected@edef\rootb{\MakeUppercase{\@nameauth@Root{#2}}}%
298         \fi

```

Use non-breaking spaces and commas as desired.

```

299         \edef\Space{\space}%
300         \edef\SpaceX{\space}%
301         \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
302         \if@nameauth@NBSPX\edef\SpaceX{\nobreakspace}\fi
303         \unless\ifx\arga\@empty
304         \if@nameauth@AlwaysComma
305         \edef\Space{,\space}%
306         \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
307         \fi
308         \if@nameauth@ShowComma
309         \edef\Space{,\space}%
310         \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
311         \fi
312         \if@nameauth@NoComma
313         \edef\Space{\space}%
314         \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
315         \fi
316         \fi

```

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

```

317 \let\SNN\rootb%
318 \ifx\arga\@empty
319 \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.

```

320 \let\FNN\suffb%
321 \let\SNN\rootb%
322 \@nameauth@NonWest{\csb#4}%
323 \else

```

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

```

324 \ifx\suffb\@empty
325 \let\FNN\argc%
326 \let\SNN\rootb%
327 \@nameauth@NonWest{\csbc#4}%
328 \else
329 \let\FNN\argc%
330 \let\SNN\rootb%
331 \@nameauth@NonWest{\csb#4}%
332 \fi
333 \fi
334 \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.

```

335 \ifx\argc\@empty
336 \let\FNN\arga%
337 \else
338 \let\FNN\argc%
339 \fi
340 \unless\ifx\suffb\@empty
341 \def\SNN{\rootb\Space\suffb}%
342 \if@nameauth@ShortSNN\let\SNN\rootb\fi
343 \fi
344 \@nameauth@West{\csab#4}%
345 \fi
346 \fi
347 }

```

`\@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.

```

348 \newcommand*\@nameauth@NonWest[1]
349 {%
350 \if@nameauth@Lock
351 \unless\ifcsname#1\endcsname
352 \@nameauth@FirstFormattrue%
353 \fi
354 \if@nameauth@InAKA
355 \if@nameauth@AltAKA
356 \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi

```

```

357     \@nameauth@FullNamefalse%
358     \@nameauth@FirstNametrue%
359     \else
360     \@nameauth@FullNametrue%
361     \@nameauth@FirstNamefalse%
362     \fi
363 \else
364     \unless\ifcsname#1\endcsname
365     \@nameauth@FullNametrue%
366     \@nameauth@FirstNamefalse%
367     \fi
368 \fi
369 \if@nameauth@FirstName
370     \@nameauth@FullNamefalse%
371 \fi
372 \ifx\FNN\@empty
373     \@nameauth@Hook{\SNN}%
374 \else
375     \if@nameauth@FullName
376     \if@nameauth@RevThis
377     \@nameauth@Hook{\FNN\Space\SNN}%
378     \else
379     \@nameauth@Hook{\SNN\Space\FNN}%
380     \fi
381 \else
382     \if@nameauth@FirstName
383     \if@nameauth@EastFN
384     \@nameauth@Hook{\FNN}%
385     \else
386     \@nameauth@Hook{\SNN}%
387     \fi
388     \else
389     \@nameauth@Hook{\SNN}%
390     \fi
391 \fi
392 \fi
393 \unless\ifcsname#1\endcsname
394     \unless\if@nameauth@InAKA\csgdef{#1}{-}\fi
395 \fi
396 \@nameauth@FullNamefalse%
397 \@nameauth@FirstNamefalse%
398 \fi
399 }

```

`\@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.

```

400 \newcommand*\@nameauth@West [1]
401 {%
402     \if@nameauth@Lock
403     \unless\ifcsname#1\endcsname
404     \@nameauth@FirstFormattrue%
405     \fi
406     \if@nameauth@InAKA
407     \if@nameauth@AltAKA
408     \@nameauth@FullNamefalse%

```

```

409     \@nameauth@FirstNametrue%
410   \else
411     \@nameauth@FullNametrue%
412     \@nameauth@FirstNamefalse%
413   \fi
414 \else
415   \unless\ifcsname#1\endcsname
416     \@nameauth@FullNametrue%
417     \@nameauth@FirstNamefalse%
418   \fi
419 \fi
420 \if@nameauth@FirstName
421   \@nameauth@FullNamefalse%
422 \fi
423 \if@nameauth@FullName
424   \if@nameauth@RevThis
425     \@nameauth@Hook{\SNN\SpaceX\FNN}%
426   \else
427     \if@nameauth@RevThisComma
428       \edef\RevSpace{,\SpaceX}%
429       \@nameauth@Hook{\SNN\RevSpace\FNN}%
430     \else
431       \@nameauth@Hook{\FNN\SpaceX\SNN}%
432     \fi
433   \fi
434 \else
435   \if@nameauth@FirstName
436     \@nameauth@Hook{\FNN}%
437   \else
438     \@nameauth@Hook{\rootb}%
439   \fi
440 \fi
441 \unless\ifcsname#1\endcsname
442   \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
443 \fi
444 \@nameauth@FullNamefalse%
445 \@nameauth@FirstNamefalse%
446 \fi
447 }

```

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.

```

448 \newcommand*\@nameauth@Hook[1]
449 {%
450   \if@nameauth@Lock
451     \@nameauth@InHooktrue%
452     \protected@edef\test{#1}%
453     \expandafter\@nameauth@TestDot\expandafter{\test}%
454     \if@nameauth@InAKA
455       \if@nameauth@AlwaysFormat
456         \@nameauth@FirstFormattrue%
457       \else
458         \unless\if@nameauth@AKAFormat

```

```

459     \@nameauth@FirstFormatfalse\fi
460 \fi
461 \if@nameauth@MainFormat
462     \if@nameauth@FirstFormat
463         \bgroup\NamesFormat{#1}\egroup%
464     \else
465         \bgroup\MainNameHook{#1}\egroup%
466     \fi
467 \else
468     \if@nameauth@FirstFormat
469         \bgroup\FrontNamesFormat{#1}\egroup%
470     \else
471         \bgroup\FrontNameHook{#1}\egroup%
472     \fi
473 \fi
474 \else
475     \if@nameauth@AlwaysFormat
476         \@nameauth@FirstFormattrue%
477     \fi
478     \if@nameauth@MainFormat
479         \if@nameauth@FirstFormat
480             \bgroup\NamesFormat{#1}\egroup%
481         \else
482             \bgroup\MainNameHook{#1}\egroup%
483         \fi
484     \else
485         \if@nameauth@FirstFormat
486             \bgroup\FrontNamesFormat{#1}\egroup%
487         \else
488             \bgroup\FrontNameHook{#1}\egroup%
489         \fi
490     \fi
491 \fi
492 \@nameauth@FirstFormatfalse%
493 \@nameauth@InHookfalse%
494 \fi
495 }

```

Indexing Internals

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

```

496 \newcommand*\@nameauth@Index[2]
497 {%
498     \def\cseq{#1}%
499     \let\ex\expandafter%
500     \ifcsname\cseq!TAG\endcsname
501         \protected@edef\Tag{\csname#1!TAG\endcsname}%
502         \ex\def\ex\ShortTag\ex{\ex\@nameauth@TrimTag\ex{\Tag}}%
503     \fi
504     \if@nameauth@DoIndex
505         \ifcsname\cseq!TAG\endcsname
506             \ifcsname\cseq!PRE\endcsname
507                 \if@nameauth@Xref%
508                     \index%
509                     {\csname\cseq!PRE\endcsname#2\ShortTag}%
510                 \else

```

```

511         \index%
512         {\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
513         \fi
514     \else
515         \if@nameauth@Xref
516             \index{#2\ShortTag}%
517         \else
518             \index{#2\csname\cseq!TAG\endcsname}%
519         \fi
520     \fi
521 \else
522     \ifcsname\cseq!PRE\endcsname
523         \index{\csname\cseq!PRE\endcsname#2}%
524     \else
525         \index{#2}%
526     \fi
527 \fi
528 \fi
529 }

```

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

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

3.5 User Interface Macros

Syntactic Formatting — Capitalization

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

```
531 \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.

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

`\CapName` Capitalize entire required name. `\CapThis` overrides this.

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

`\AllCapsInactive` Turn off global surname capitalization. `\CapThis` overrides this.

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

`\AllCapsActive` Turn on global surname capitalization. `\CapThis` overrides this.

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

Syntactic Formatting — Reversing

`\RevName` Reverse name order.

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

`\ReverseInactive` Turn off global name reversing.

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

`\ReverseActive` Turn on global name reversing.

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

`\ForceFN` Force the printing of an Eastern forename in the text, but only when using the “short name” macro `\FName` and the S-modifier.

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

Syntactic Formatting — Reversing with Commas

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

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

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

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

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

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

Alternate Syntactic Formatting

`\AltFormatActive` Turn on alternate formatting.

```
547 \newcommand*\AltFormatActive{%
548   \global\@nameauth@AltFormattrue%
549   \global\@nameauth@DoAlttrue%
550 }
```

`\AltFormatActive*` Turn on alternate formatting.

```
551 \WithSuffix{\newcommand*}\AltFormatActive*{%
552   \global\@nameauth@AltFormattrue%
553   \global\@nameauth@DoAltfalse%
554 }
```

`\AltFormatInactive` Turn off alternate formatting.

```
555 \newcommand*\AltFormatInactive{%
556   \global\@nameauth@AltFormatfalse%
557   \global\@nameauth@DoAltfalse%
558 }
```

`\AltOn` Locally turn on alternate formatting.

```
559 \newcommand*\AltOn{%
560   \if@nameauth@InHook
561     \if@nameauth@AltFormat\@nameauth@DoAlttrue\fi
562   \fi
563 }
```

`\AltOff` Locally turn off alternate formatting.

```
564 \newcommand*\AltOff{%
565   \if@nameauth@InHook
566     \if@nameauth@AltFormat\@nameauth@DoAltfalse\fi
567   \fi
568 }
```


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

```

569 \newcommand*\AltCaps[1]{%
570   \if@nameauth@InHook
571     \if@nameauth@DoCaps\MakeUppercase{#1}\else#1\fi
572   \else#1%
573   \fi
574 }
```

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

```

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

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

```

577 \newcommand*\textUC[1]{%
578   \if@nameauth@DoAlt\MakeUppercase{#1}\else#1\fi}
```

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

```

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

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

```

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

Syntactic Formatting — Affixes

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

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

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

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

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

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

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

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

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

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

Typographic Formatting — Main Versus Front Matter

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

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

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

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

Typographic Formatting — First / Subsequent Reference

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

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

`\SubvertThis` Have the naming engine `\@nameauth@Name` call `\SubvertName` internally.
591 `\newcommand*\SubvertThis{\@nameauth@Subverttrue}`

`\ForceName` Set `\@nameauth@FirstFormat` to be true even for subsequent name uses. Works for one name only.
592 `\newcommand*\ForceName{\@nameauth@FirstFormattrue}`

Name Occurrence Tweaks

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

`\GlobalNames` `\GlobalNames` sets `@nameauth@LocalNames` false. This restores the default behavior of `\ForgetName` and `\SubvertName`.
594 `\newcommand*\GlobalNames{\global\@nameauth@LocalNamesfalse}`

Index Operations

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

`\SkipIndex` Turn off the next instance of indexing in `\Name`, `\FName`, and starred forms.
596 `\newcommand*\SkipIndex{\@nameauth@SkipIndextrue}`

`\JustIndex` Makes the next call to `\Name`, `\FName`, and starred forms act like `\IndexName`. Overrides `\SkipIndex`.
597 `\newcommand*\JustIndex{\@nameauth@JustIndextrue}`

`\IndexActive` Turn on global indexing of names.
598 `\newcommand*\IndexActive{\@nameauth@DoIndextrue}`

`\IndexActual` Change the “actual” character from the default.
599 `\newcommand*\IndexActual[1]`
600 `{\global\renewcommand*\@nameauth@Actual{#1}}`

`\SeeAlso` Change the type of cross-reference from a *see* reference to a *see also* reference. Works once per xref, unless one uses `\Include*`, in which case, take care!
601 `\newcommand*\SeeAlso{\@nameauth@SeeAlsottrue}`

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.
602 `\newcommand*\NameParser`
603 `{%`
604 `\if@nameauth@InHook`
605 `\let\SNN\rootb%`
606 `\ifx\arga\@empty`

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

607 `\ifx\argc\@empty`
608 `\let\FNN\suffb%`
609 `\else`
610 `\let\FNN\argc%`
611 `\fi`
612 `\ifx\suffb\@empty`

Mononym case

```
613     \ifx\FNN\@empty
614         \SNN%
615     \else
```

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

```
616         \if@nameauth@FullName%
617         \if@nameauth@RevThis
618             \FNN\Space\SNN%
619         \else
620             \SNN\Space\FNN%
621         \fi
622     \else
623         \if@nameauth@FirstName
624             \if@nameauth@EastFN
625                 \FNN%
626             \else
627                 \SNN%
628             \fi
629         \else
630             \SNN%
631         \fi
632     \fi
633 \fi
634 \else
```

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

```
635     \if@nameauth@FullName
636     \if@nameauth@RevThis
637         \FNN\Space\SNN%
638     \else
639         \SNN\Space\FNN%
640     \fi
641 \else
642     \if@nameauth@FirstName
643         \if@nameauth@EastFN
644             \FNN%
645         \else
646             \SNN%
647         \fi
648     \else
649         \SNN%
650     \fi
651 \fi
652 \fi
653 \else
```

Western name with name reversing and suffixes

```
654     \ifx\argc\@empty
655         \let\FNN\arga%
656     \else
657         \let\FNN\argc%
658     \fi
659 \unless\ifx\suffb\@empty
660     \def\SNN{\rootb\Space\suffb}%
661     \if@nameauth@ShortSNN\let\SNN\rootb\fi%
662 \fi
```

```

663     \if@nameauth@FullName
664     \if@nameauth@RevThis
665     \SNN\SpaceX\FNN%
666     \else
667     \if@nameauth@RevThisComma
668     \SNN\RevSpace\FNN%
669     \else
670     \FNN\SpaceX\SNN%
671     \fi
672     \fi
673     \else
674     \if@nameauth@FirstName
675     \FNN%
676     \else
677     \let\SNN\rootb%
678     \SNN%
679     \fi
680     \fi
681     \fi
682     \fi
683 }

```

Traditional Naming Interface

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

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

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

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

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

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

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

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

Index Operations

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

```

690 \newcommandx*\IndexName[3][1=\@empty, 3=\@empty]
691 {%
692   \protected@edef\arga{\trim@spaces{#1}}%
693   \protected@edef\rootb{\@nameauth@Root{#2}}%
694   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
695   \protected@edef\argc{\trim@spaces{#3}}%
696   \def\csb{\@nameauth@Clean{#2}}%
697   \def\csbc{\@nameauth@Clean{#2,#3}}%
698   \def\csab{\@nameauth@Clean{#1!#2}}%

```

Test for malformed input.

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

```

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

```

700 \ifx\arga\@empty
701 \ifx\argc\@empty
702 \ifcsname\csb!PN\endcsname
703 \if@nameauth@Verbose
704 \PackageWarning{nameauth}%
705 {macro \IndexName: XRef: #2 exists}%
706 \fi
707 \else
708 \ifx\suffb\@empty

```

mononym or Eastern / ancient name, new syntax

```

709 \@nameauth@Index{\csb}{\rootb}%
710 \else
711 \@nameauth@Index{\csb}{\rootb\space\suffb}%
712 \fi
713 \fi
714 \else
715 \ifx\suffb\@empty
716 \ifcsname\csbc!PN\endcsname
717 \if@nameauth@Verbose
718 \PackageWarning{nameauth}%
719 {macro \IndexName: XRef: #2 #3 exists}%
720 \fi
721 \else

```

Eastern or ancient name, older syntax

```

722 \@nameauth@Index{\csbc}{\rootb\space\argc}%
723 \fi
724 \else
725 \ifcsname\csb!PN\endcsname
726 \if@nameauth@Verbose
727 \PackageWarning{nameauth}%
728 {macro \IndexName: XRef: #2 exists}%
729 \fi
730 \else

```

Eastern or ancient name, new syntax, alternate name ignored

```

731 \@nameauth@Index{\csb}{\rootb\space\suffb}%
732 \fi
733 \fi
734 \fi
735 \else
736 \ifcsname\csab!PN\endcsname
737 \if@nameauth@Verbose
738 \PackageWarning{nameauth}%
739 {macro \IndexName: XRef: #1 #2 exists}%
740 \fi
741 \else

```

Western name, without and with affix

```

742 \ifx\suffb\@empty
743 \@nameauth@Index{\csab}%
744 {\rootb,\space\arga}%

```

```

745     \else
746         \@nameauth@Index{\csab}%
747         {\rootb,\space\arga,\space\suffb}%
748     \fi
749 \fi
750 \fi
751 }

```

`\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.

```

752 \newcommand*\IndexRef[4][1=\@empty, 3=\@empty]
753 {%
754 \protected@edef\arga{\trim@spaces{#1}}%
755 \protected@edef\rootb{\@nameauth@Root{#2}}%
756 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
757 \protected@edef\argc{\trim@spaces{#3}}%
758 \protected@edef\target{#4}%
759 \def\csb{\@nameauth@Clean{#2}}%
760 \def\csbc{\@nameauth@Clean{#2,#3}}%
761 \def\csab{\@nameauth@Clean{#1!#2}}%
762 \let\ex\expandafter%

```

Test for malformed input.

```

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

```

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

```

765 \ifx\arga\@empty
766 \ifx\argc\@empty
767 \ifcsname\csb!PN\endcsname
768 \if@nameauth@Verbose
769 \PackageWarning{nameauth}%
770 {macro \IndexRef: XRef: #2 exists}%
771 \fi
772 \else
773 \ifx\suffb\@empty

```

mononym or Eastern / ancient name, new syntax

```

774 \if@nameauth@SeeAlso
775 \@nameauth@Index{\csb}{\rootb|seealso{\target}}%
776 \else
777 \@nameauth@Index{\csb}{\rootb|see{\target}}%
778 \fi
779 \else
780 \if@nameauth@SeeAlso
781 \@nameauth@Index{\csb}{\rootb\space\suffb|seealso{\target}}%
782 \else
783 \@nameauth@Index{\csb}{\rootb\space\suffb|see{\target}}%
784 \fi
785 \fi
786 \csgdef{\csb!PN}{}%
787 \fi
788 \else
789 \ifx\suffb\@empty
790 \ifcsname\csbc!PN\endcsname
791 \if@nameauth@Verbose

```

```

792         \PackageWarning{nameauth}%
793         {macro \IndexRef: XRef: #2 #3 exists}%
794     \fi
795     \else

```

Eastern or ancient name, older syntax

```

796     \if@nameauth@SeeAlso
797     \@nameauth@Index{\csbc}%
798     {\rootb\space\argc|seealso{\target}}%
799     \else
800     \@nameauth@Index{\csbc}%
801     {\rootb\space\argc|see{\target}}%
802     \fi
803     \csgdef{\csbc!PN}{}%
804 \fi
805 \else
806     \ifcsname\csb!PN\endcsname
807     \if@nameauth@Verbose
808     \PackageWarning{nameauth}%
809     {macro \IndexRef: XRef: #2 exists}%
810     \fi
811     \else

```

Eastern or ancient name, new syntax, alternate name ignored

```

812     \if@nameauth@SeeAlso
813     \@nameauth@Index{\csb}%
814     {\rootb\space\suffb|seealso{\target}}%
815     \else
816     \@nameauth@Index{\csb}%
817     {\rootb\space\suffb|see{\target}}%
818     \fi
819     \csgdef{\csb!PN}{}%
820 \fi
821 \fi
822 \fi
823 \else
824     \ifcsname\csab!PN\endcsname
825     \if@nameauth@Verbose
826     \PackageWarning{nameauth}%
827     {macro \IndexRef: XRef: #1 #2 exists}%
828     \fi
829     \else

```

Western name, without and with affix

```

830     \ifx\suffb\@empty
831     \if@nameauth@SeeAlso
832     \@nameauth@Index{\csab}%
833     {\rootb,\space\argc|seealso{\target}}%
834     \else
835     \@nameauth@Index{\csab}%
836     {\rootb,\space\argc|see{\target}}%
837     \fi
838     \else
839     \if@nameauth@SeeAlso
840     \@nameauth@Index{\csab}%
841     {\rootb,\space\argc,\space\suffb|seealso{\target}}%
842     \else

```

```

843         \@nameauth@Index{\csab}%
844         {\rootb,\space\arga,\space\suffb|see{\target}}%
845         \fi
846     \fi
847     \csgdef{\csab!PN}{}%
848 \fi
849 \fi
850 \@nameauth@SeeAlsofalse%
851 \@nameauth@Xreffalse%
852 }

```

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

```

853 \newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]
854 {%
855     \protected@edef\arga{\trim@spaces{#1}}%
856     \protected@edef\argc{\trim@spaces{#3}}%
857     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
858     \def\csb{\@nameauth@Clean{#2}}%
859     \def\csbc{\@nameauth@Clean{#2,#3}}%
860     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

861 \@nameauth@Error{#2}{macro \string\ExcludeName}%
862 \ifx\arga\@empty
863     \ifx\argc\@empty
864         \if@nameauth@Verbose
865             \ifcsname\csb!MN\endcsname
866                 \PackageWarning{nameauth}%
867                 {macro \ExcludeName: Reference: #2 exists}%
868             \fi
869             \ifcsname\csb!NF\endcsname
870                 \PackageWarning{nameauth}%
871                 {macro \ExcludeName: Reference: #2 exists}%
872             \fi
873         \fi
874         \ifcsname\csb!PN\endcsname
875             \if@nameauth@Verbose
876                 \PackageWarning{nameauth}%
877                 {macro \ExcludeName: Xref: #2 exists}%
878             \fi
879         \else
880             \csgdef{\csb!PN}{!}%
881         \fi
882     \else
883         \ifx\suffb\@empty
884             \if@nameauth@Verbose
885                 \ifcsname\csbc!MN\endcsname
886                     \PackageWarning{nameauth}%
887                     {macro \ExcludeName: Reference: #2 #3 exists}%
888                 \fi
889                 \ifcsname\csbc!NF\endcsname
890                     \PackageWarning{nameauth}%
891                     {macro \ExcludeName: Reference: #2 #3 exists}%
892                 \fi
893             \fi
894             \csgdef{\csbc!PN}{!}%

```



```

895     \ifcsname\csbc!PN\endcsname
896     \if@nameauth@Verbose
897     \PackageWarning{nameauth}%
898     {macro \ExcludeName: Xref: #2 exists}%
899     \fi
900   \else
901     \csgdef{\csbc!PN}{!}%
902   \fi
903 \else
904   \if@nameauth@Verbose
905     \ifcsname\csb!MN\endcsname
906     \PackageWarning{nameauth}%
907     {macro \ExcludeName: Reference: #2 exists}%
908   \fi
909   \ifcsname\csb!NF\endcsname
910   \PackageWarning{nameauth}%
911   {macro \ExcludeName: Reference: #2 exists}%
912   \fi
913 \fi
914 \ifcsname\csb!PN\endcsname
915   \if@nameauth@Verbose
916   \PackageWarning{nameauth}%
917   {macro \ExcludeName: Xref: #2 exists}%
918   \fi
919   \else
920     \csgdef{\csb!PN}{!}%
921   \fi
922 \fi
923 \fi
924 \else
925   \if@nameauth@Verbose
926     \ifcsname\csab!MN\endcsname
927     \PackageWarning{nameauth}%
928     {macro \ExcludeName: Reference: #1 #2 exists}%
929   \fi
930   \ifcsname\csab!NF\endcsname
931   \PackageWarning{nameauth}%
932   {macro \ExcludeName: Reference: #1 #2 exists}%
933   \fi
934 \fi
935 \ifcsname\csab!PN\endcsname
936   \if@nameauth@Verbose
937   \PackageWarning{nameauth}%
938   {macro \ExcludeName: Xref: #2 exists}%
939   \fi
940   \else
941     \csgdef{\csab!PN}{!}%
942   \fi
943 \fi
944 }

```

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

```

945 \newcommand*\IncludeName[3][1=\@empty, 3=\@empty]
946 {%
947   \protected@edef\arga{\trim@spaces{#1}}%
948   \protected@edef\argc{\trim@spaces{#3}}%

```

```

949 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
950 \def\csb{\@nameauth@Clean{#2}}%
951 \def\csbc{\@nameauth@Clean{#2,#3}}%
952 \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

953 \@nameauth@Error{#2}{macro \string\IncludeName}%
954 \ifx\arga\@empty
955   \ifx\argc\@empty
956     \ifcsname\csb!PN\endcsname
957       \edef\testex{\csname\csb!PN\endcsname}%
958       \unless\ifx\testex\@empty\global\csundef{\csb!PN}\fi
959     \fi
960   \else
961     \ifx\suffb\@empty
962       \ifcsname\csbc!PN\endcsname
963         \edef\testex{\csname\csbc!PN\endcsname}%
964         \unless\ifx\testex\@empty\global\csundef{\csbc!PN}\fi
965       \fi
966     \else
967       \ifcsname\csb!PN\endcsname
968         \edef\testex{\csname\csb!PN\endcsname}%
969         \unless\ifx\testex\@empty\global\csundef{\csb!PN}\fi
970       \fi
971     \fi
972   \fi
973 \else
974   \ifcsname\csab!PN\endcsname
975     \edef\testex{\csname\csab!PN\endcsname}%
976     \unless\ifx\testex\@empty\global\csundef{\csab!PN}\fi
977   \fi
978 \fi
979 }

```

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

```

980 \WithSuffix{\newcommand*}\IncludeName*[3][1=\@empty, 3=\@empty]
981 {%
982   \protected@edef\arga{\trim@spaces{#1}}%
983   \protected@edef\argc{\trim@spaces{#3}}%
984   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
985   \def\csb{\@nameauth@Clean{#2}}%
986   \def\csbc{\@nameauth@Clean{#2,#3}}%
987   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

988 \@nameauth@Error{#2}{macro \string\IncludeName*}%
989 \ifx\arga\@empty
990   \ifx\argc\@empty
991     \global\csundef{\csb!PN}%
992   \else
993     \ifx\suffb\@empty
994       \global\csundef{\csbc!PN}%
995     \else
996       \global\csundef{\csb!PN}%
997     \fi
998   \fi

```

```

999  \else
1000  \global\csundef{\csab!PN}%
1001  \fi
1002 }

```

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

```

1003 \newcommand*\PretagName[4][1=\@empty, 3=\@empty]
1004 {%
1005  \protected@edef\arga{\trim@spaces{#1}}%
1006  \protected@edef\argc{\trim@spaces{#3}}%
1007  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1008  \def\csb{\@nameauth@Clean{#2}}%
1009  \def\csbc{\@nameauth@Clean{#2,#3}}%
1010  \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1011  \@nameauth@Error{#2}{macro \string\PretagName}%
1012  \ifx\arga\@empty
1013    \ifx\argc\@empty
1014      \ifcsname\csb!PN\endcsname
1015        \if@nameauth@Verbose
1016          \PackageWarning{nameauth}%
1017          {macro \PretagName: tagging xref: #2}%
1018        \fi
1019      \fi
1020      \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
1021    \else
1022      \ifx\suffb\@empty
1023        \ifcsname\csbc!PN\endcsname
1024          \if@nameauth@Verbose
1025            \PackageWarning{nameauth}%
1026            {macro \PretagName: tagging xref: #2 #3}%
1027          \fi
1028        \fi
1029        \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
1030      \else
1031        \ifcsname\csb!PN\endcsname
1032          \if@nameauth@Verbose
1033            \PackageWarning{nameauth}%
1034            {macro \PretagName: tagging xref: #2}%
1035          \fi
1036        \fi
1037        \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
1038      \fi
1039    \fi
1040  \else
1041    \ifcsname\csab!PN\endcsname
1042      \if@nameauth@Verbose
1043        \PackageWarning{nameauth}%
1044        {macro \PretagName: tagging xref: #1 #2}%
1045      \fi
1046    \fi
1047    \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
1048  \fi
1049 }

```

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

```
1050 \newcommandx*\TagName [4] [1=\@empty, 3=\@empty]
1051 {%
1052   \protected@edef\arga{\trim@spaces{#1}}%
1053   \protected@edef\argc{\trim@spaces{#3}}%
1054   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1055   \def\csb{\@nameauth@Clean{#2}}%
1056   \def\csbc{\@nameauth@Clean{#2,#3}}%
1057   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1058 \@nameauth@Error{#2}{macro \string\TagName}%
1059 \ifx\arga\@empty
1060   \ifx\argc\@empty
1061     \ifcsname\csb!PN\endcsname
1062       \if@nameauth@Verbose
1063         \PackageWarning{nameauth}%
1064         {macro \TagName: not tagging xref: #2}%
1065       \fi
1066     \else
1067       \csgdef{\csb!TAG}{#4}%
1068     \fi
1069   \else
1070     \ifx\suffb\@empty
1071       \ifcsname\csbc!PN\endcsname
1072         \if@nameauth@Verbose
1073           \PackageWarning{nameauth}%
1074           {macro \TagName: not tagging xref: #2 #3}%
1075         \fi
1076       \else
1077         \csgdef{\csbc!TAG}{#4}%
1078       \fi
1079     \else
1080       \ifcsname\csb!PN\endcsname
1081         \if@nameauth@Verbose
1082           \PackageWarning{nameauth}%
1083           {macro \TagName: not tagging xref: #2}%
1084         \fi
1085       \else
1086         \csgdef{\csb!TAG}{#4}%
1087       \fi
1088     \fi
1089   \fi
1090 \else
1091   \ifcsname\csab!PN\endcsname
1092     \if@nameauth@Verbose
1093       \PackageWarning{nameauth}%
1094       {macro \TagName: not tagging xref: #1 #2}%
1095     \fi
1096   \else
1097     \csgdef{\csab!TAG}{#4}%
1098   \fi
1099 \fi
1100 }
```

`\UntagName` This deletes an index tag.

```
1101 \newcommandx*\UntagName[3][1=\@empty, 3=\@empty]
1102 {%
1103   \protected@edef\arga{\trim@spaces{#1}}%
1104   \protected@edef\argc{\trim@spaces{#3}}%
1105   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1106   \def\csb{\@nameauth@Clean{#2}}%
1107   \def\csbc{\@nameauth@Clean{#2,#3}}%
1108   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1109 \@nameauth@Error{#2}{macro \string\UntagName}%
1110 \ifx\arga\@empty
1111   \ifx\argc\@empty
1112     \global\csundef{\csb!TAG}%
1113   \else
1114     \ifx\suffb\@empty
1115       \global\csundef{\csbc!TAG}%
1116     \else
1117       \global\csundef{\csb!TAG}%
1118     \fi
1119   \fi
1120 \else
1121   \global\csundef{\csab!TAG}%
1122 \fi
1123 }
```

Name Info Data Set: “Text Tags”

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

```
1124 \newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
1125 {%
1126   \protected@edef\arga{\trim@spaces{#1}}%
1127   \protected@edef\argc{\trim@spaces{#3}}%
1128   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1129   \def\csb{\@nameauth@Clean{#2}}%
1130   \def\csbc{\@nameauth@Clean{#2,#3}}%
1131   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1132 \@nameauth@Error{#2}{macro \string\NameAddInfo}%
1133 \ifx\arga\@empty
1134   \ifx\argc\@empty
1135     \csgdef{\csb!DB}{#4}%
1136   \else
1137     \ifx\Suff\@empty
1138       \csgdef{\csbc!DB}{#4}%
1139     \else
1140       \csgdef{\csb!DB}{#4}%
1141     \fi
1142   \fi
1143 \else
1144   \csgdef{\csab!DB}{#4}%
1145 \fi
1146 }
```

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

```
1147 \newcommandx*\NameQueryInfo[3][1=\@empty, 3=\@empty]
1148 {%
1149   \protected@edef\arga{\trim@spaces{#1}}%
1150   \protected@edef\argc{\trim@spaces{#3}}%
1151   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1152   \def\csb{\@nameauth@Clean{#2}}%
1153   \def\csbc{\@nameauth@Clean{#2,#3}}%
1154   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1155   \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
1156   \ifx\arga\@empty
1157     \ifx\argc\@empty
1158       \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1159     \else
1160       \ifx\Suff\@empty
1161         \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
1162       \else
1163         \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1164       \fi
1165     \fi
1166   \else
1167     \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
1168   \fi
1169 }
```

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

```
1170 \newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
1171 {%
1172   \protected@edef\arga{\trim@spaces{#1}}%
1173   \protected@edef\argc{\trim@spaces{#3}}%
1174   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1175   \def\csb{\@nameauth@Clean{#2}}%
1176   \def\csbc{\@nameauth@Clean{#2,#3}}%
1177   \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1178   \@nameauth@Error{#2}{macro \string\NameClearInfo}%
1179   \ifx\arga\@empty
1180     \ifx\argc\@empty
1181       \global\csundef{\csb!DB}%
1182     \else
1183       \ifx\Suff\@empty
1184         \global\csundef{\csbc!DB}%
1185       \else
1186         \global\csundef{\csb!DB}%
1187       \fi
1188     \fi
1189   \else
1190     \global\csundef{\csab!DB}%
1191   \fi
1192 }
```

Name Decisions

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

```
1193 \newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
1194 {%
1195   \protected@edef\arga{\trim@spaces{#1}}%
1196   \protected@edef\argc{\trim@spaces{#3}}%
1197   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1198   \def\csb{\@nameauth@Clean{#2}}%
1199   \def\csbc{\@nameauth@Clean{#2,#3}}%
1200   \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
1201   \@nameauth@Error{#2}{macro \string\IfMainName}%
1202   \ifx\arga\@empty
1203     \ifx\argc\@empty
1204       \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1205     \else
1206       \ifx\suffb\@empty
1207         \ifcsname\csbc!MN\endcsname{#4}\else{#5}\fi
1208       \else
1209         \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1210       \fi
1211     \fi
1212   \else
1213     \ifcsname\csab!MN\endcsname{#4}\else{#5}\fi
1214   \fi
1215 }
```

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

```
1216 \newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
1217 {%
1218   \protected@edef\arga{\trim@spaces{#1}}%
1219   \protected@edef\argc{\trim@spaces{#3}}%
1220   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1221   \def\csb{\@nameauth@Clean{#2}}%
1222   \def\csbc{\@nameauth@Clean{#2,#3}}%
1223   \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
1224   \@nameauth@Error{#2}{macro \string\IfFrontName}%
1225   \ifx\arga\@empty
1226     \ifx\argc\@empty
1227       \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1228     \else
1229       \ifx\suffb\@empty
1230         \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
1231       \else
1232         \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1233       \fi
1234     \fi
1235   \else
1236     \ifcsname\csab!NF\endcsname{#4}\else{#5}\fi
1237   \fi
1238 }
```

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

```

1239 \newcommand\IfAKA[6][1=\@empty, 3=\@empty]
1240 {%
1241   \protected@edef\arga{\trim@spaces{#1}}%
1242   \protected@edef\argc{\trim@spaces{#3}}%
1243   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1244   \def\csb{\@nameauth@Clean{#2}}%
1245   \def\csbc{\@nameauth@Clean{#2,#3}}%
1246   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1247   \@nameauth@Error{#2}{macro \string\IfAKA}%
1248   \ifx\arga\@empty
1249     \ifx\argc\@empty
1250       \ifcsname\csb!PN\endcsname
1251         \edef\testex{\csname\csb!PN\endcsname}%
1252         \ifx\testex\@empty{#4}\else{#6}\fi
1253       \else{#5}\fi
1254     \else
1255       \ifx\suffb\@empty
1256         \ifcsname\csbc!PN\endcsname
1257           \edef\testex{\csname\csbc!PN\endcsname}%
1258           \ifx\testex\@empty{#4}\else{#6}\fi
1259         \else{#5}\fi
1260       \else
1261         \ifcsname\csb!PN\endcsname
1262           \edef\testex{\csname\csb!PN\endcsname}%
1263           \ifx\testex\@empty{#4}\else{#6}\fi
1264         \else{#5}\fi
1265       \fi
1266     \fi
1267   \else
1268     \ifcsname\csab!PN\endcsname
1269       \edef\testex{\csname\csab!PN\endcsname}%
1270       \ifx\testex\@empty{#4}\else{#6}\fi
1271     \else{#5}\fi
1272   \fi
1273 }

```

Changing Name Decisions

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

```

1274 \newcommand*\ForgetName[3][1=\@empty, 3=\@empty]
1275 {%
1276   \protected@edef\arga{\trim@spaces{#1}}%
1277   \protected@edef\argc{\trim@spaces{#3}}%
1278   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1279   \def\csb{\@nameauth@Clean{#2}}%
1280   \def\csbc{\@nameauth@Clean{#2,#3}}%
1281   \def\csab{\@nameauth@Clean{#1!#2}}%
1282   \@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).

```

1283 \ifx\arga\@empty
1284 \ifx\argc\@empty
1285 \if@nameauth@LocalNames
1286 \if@nameauth@MainFormat
1287 \global\csundef{\csb!MN}%
1288 \else
1289 \global\csundef{\csb!NF}%
1290 \fi
1291 \else
1292 \global\csundef{\csb!MN}%
1293 \global\csundef{\csb!NF}%
1294 \fi
1295 \else
1296 \ifx\suffb\@empty
1297 \if@nameauth@LocalNames
1298 \if@nameauth@MainFormat
1299 \global\csundef{\csbc!MN}%
1300 \else
1301 \global\csundef{\csbc!NF}%
1302 \fi
1303 \else
1304 \global\csundef{\csbc!MN}%
1305 \global\csundef{\csbc!NF}%
1306 \fi
1307 \else
1308 \if@nameauth@LocalNames
1309 \if@nameauth@MainFormat
1310 \global\csundef{\csb!MN}%
1311 \else
1312 \global\csundef{\csb!NF}%
1313 \fi
1314 \else
1315 \global\csundef{\csb!MN}%
1316 \global\csundef{\csb!NF}%
1317 \fi
1318 \fi
1319 \fi
1320 \else
1321 \if@nameauth@LocalNames
1322 \if@nameauth@MainFormat
1323 \global\csundef{\csab!MN}%
1324 \else
1325 \global\csundef{\csab!NF}%
1326 \fi
1327 \else
1328 \global\csundef{\csab!MN}%
1329 \global\csundef{\csab!NF}%
1330 \fi
1331 \fi
1332 }

```

```

\SubvertName This defines a control sequence to force a “subsequent use.”
1333 \newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
1334 {%
1335   \protected@edef\arga{\trim@spaces{#1}}%
1336   \protected@edef\argc{\trim@spaces{#3}}%
1337   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1338   \def\csb{\@nameauth@Clean{#2}}%
1339   \def\csbc{\@nameauth@Clean{#2,#3}}%
1340   \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them.

```
1341   \@nameauth@Error{#2}{macro \string\SubvertName}%
```

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

```

1342   \ifx\arga\@empty
1343     \ifx\argc\@empty
1344       \if@nameauth@LocalNames
1345         \if@nameauth@MainFormat
1346           \csgdef{\csb!MN}{}%
1347         \else
1348           \csgdef{\csb!NF}{}%
1349         \fi
1350       \else
1351         \csgdef{\csb!MN}{}%
1352         \csgdef{\csb!NF}{}%
1353       \fi
1354     \else
1355       \ifx\suffb\@empty
1356         \if@nameauth@LocalNames
1357           \if@nameauth@MainFormat
1358             \csgdef{\csbc!MN}{}%
1359           \else
1360             \csgdef{\csbc!NF}{}%
1361           \fi
1362         \else
1363           \csgdef{\csbc!MN}{}%
1364           \csgdef{\csbc!NF}{}%
1365         \fi
1366       \else
1367         \if@nameauth@LocalNames
1368           \if@nameauth@MainFormat
1369             \csgdef{\csb!MN}{}%
1370           \else
1371             \csgdef{\csb!NF}{}%
1372           \fi
1373         \else
1374           \csgdef{\csb!MN}{}%
1375           \csgdef{\csb!NF}{}%
1376         \fi
1377       \fi
1378     \fi
1379   \else
1380     \if@nameauth@LocalNames
1381       \if@nameauth@MainFormat

```

```

1382     \csgdef{\csab!MN}{}%
1383     \else
1384     \csgdef{\csab!NF}{}%
1385     \fi
1386     \else
1387     \csgdef{\csab!MN}{}%
1388     \csgdef{\csab!NF}{}%
1389     \fi
1390 \fi
1391 }

```

Alternate Names

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

```

1392 \newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]
1393 {%

```

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

```

1394 \unless\if@nameauth@Lock
1395 \@nameauth@Locktrue%
1396 \@nameauth@JustIndexfalse%

```

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

```

1397 \@nameauth@InAKAtrue%
1398 \@nameauth@Error{#2}{macro \string\AKA}%
1399 \@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.

```

1400 \leavevmode\hbox{%
1401 \protected@edef\argi{\trim@spaces{#1}}%
1402 \protected@edef\rooti{\@nameauth@Root{#2}}%
1403 \protected@edef\suffi{\@nameauth@Suffix{#2}}%
1404 \@nameauth@Parse[#3]{#4}[#5]{!PN}%

```

Create an index cross-reference based on the arguments.

```

1405 \unless\if@nameauth@SkipIndex
1406   \ifx\argi\@empty
1407     \ifx\suffi\@empty
1408       \IndexRef[#3]{#4}[#5]{\rooti}%
1409     \else
1410       \IndexRef[#3]{#4}[#5]{\rooti\space\suffi}%
1411     \fi
1412   \else
1413     \ifx\suffi\@empty
1414       \IndexRef[#3]{#4}[#5]{\rooti,\space\argi}%
1415     \else
1416       \IndexRef[#3]{#4}[#5]{\rooti,\space\argi,\space\suffi}%
1417     \fi
1418   \fi
1419 \fi

```

Reset all the “per name” Boolean values.

```

1420 \@nameauth@SkipIndexfalse%
1421 \@nameauth@Lockfalse%
1422 \@nameauth@InAKAfalse%
1423 \@nameauth@AltAKAfalse%

```

```

1424 \@nameauth@NBSPfalse%
1425 \@nameauth@NBSPXfalse%
1426 \@nameauth@DoCapsfalse%
1427 \@nameauth@Accentfalse%
1428 \@nameauth@AllThisfalse%
1429 \@nameauth@ShowCommafalse%
1430 \@nameauth@NoCommafalse%
1431 \@nameauth@RevThisfalse%
1432 \@nameauth@RevThisCommafalse%
1433 \@nameauth@ShortSNNfalse%
1434 \@nameauth@EastFNfalse%

```

Close the “locked” branch.

```
1435 \fi
```

Call the full stop detection.

```

1436 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
1437 }

```

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

```
1438 \WithSuffix{\newcommand*}\AKA*{\@nameauth@AltAKAtrue\AKA}
```

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

```

1439 \newcommandx*\PName [5] [1=\@empty,3=\@empty,5=\@empty]
1440 {%
1441   \@nameauth@JustIndexfalse%
1442   \if@nameauth@SkipIndex
1443     \NameauthName[#1]{#2}\space(\SkipIndex\AKA[#1]{#2}[#3]{#4}[#5])%
1444   \else
1445     \NameauthName[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
1446   \fi
1447 }

```

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

```
1448 \WithSuffix{\newcommand*}\PName*{\@nameauth@FullName>true\PName}
```

Simplified Interface

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

```

1449 \newenvironment{nameauth}{%
1450   \begingroup%
1451   \let\ex\expandafter%
1452   \csdef{<##1&##2&##3&##4>{%
1453     \protected@edef\@arga{\trim@spaces{##1}}%
1454     \protected@edef\@testb{\trim@spaces{##2}}%
1455     \protected@edef\@testd{\trim@spaces{##4}}%
1456     \@nameauth@etoksb\expandafter{##2}%
1457     \@nameauth@etoksc\expandafter{##3}%
1458     \@nameauth@etoksd\expandafter{##4}%

```

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

```
1459   \ifx\@arga\@empty
```

```

1460     \PackageError{nameauth}%
1461     {environment nameauth: Control sequence missing}%
1462     \fi
1463     \@nameauth@Error{##3}{environment nameauth}%
1464     \ifcsname\@arga\endcsname
1465     \PackageWarning{nameauth}%
1466     {environment nameauth: Shorthand macro already exists}%
1467     \fi

```

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

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

```

1468     \ifx\@testd\@empty
1469     \ifx\@testb\@empty
1470     \ex\csgdef\ex{\ex\@arga\ex}\ex{\ex\NameauthName\ex{%
1471     \the\@nameauth@etoksc}}%
1472     \ex\csgdef\ex{\ex L\ex\@arga\ex}\ex{%
1473     \ex\@nameauth@FullNametrue%
1474     \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
1475     \ex\csgdef\ex{\ex S\ex\@arga\ex}\ex{%
1476     \ex\@nameauth@FirstNametrue%
1477     \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
1478     \else

```

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

```

1479     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1480     \ex\ex\ex{\ex\ex\ex\NameauthName\ex\ex\ex[%
1481     \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
1482     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
1483     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1484     \ex\ex\ex\NameauthLName\ex\ex\ex[%
1485     \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
1486     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
1487     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1488     \ex\ex\ex\NameauthFName\ex\ex\ex[%
1489     \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
1490     \fi
1491     \else

```

Below are native Eastern names with alternates and the older syntax. Again, we have three or one use of `\expandafter` per step before the respective arguments.

```

1492     \ifx\@testb\@empty
1493     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1494     \ex\ex\ex{\ex\ex\ex\NameauthName\ex\ex\ex%
1495     \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
1496     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
1497     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1498     \ex\ex\ex\NameauthLName\ex\ex\ex%
1499     \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
1500     \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
1501     \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1502     \ex\ex\ex\NameauthFName\ex\ex\ex%
1503     \ex\the\ex\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
1504     \else

```

Here are Western names with alternates. We have three arguments to expand, so we have seven, three, and one respective use of `\expandafter`.

```

1505     \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1506     \ex\ex\ex\ex\ex\ex\ex\@arga\ex\ex\ex\ex\ex\ex\ex}%
1507     \ex\ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\NameauthName%
1508     \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb%
1509     \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1510     \the\@nameauth@etoksd]}}%
1511     \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1512     \ex\ex\ex\ex\ex\ex\ex L\ex\ex\ex\ex\ex\ex\ex\@arga%
1513     \ex\ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex\ex{%
1514     \ex\ex\ex\ex\ex\ex\ex\ex\@nameauth@FullNametrue%
1515     \ex\ex\ex\ex\ex\ex\ex\ex\NameauthLName\ex\ex\ex\ex\ex\ex\ex[%
1516     \ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb%
1517     \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1518     \the\@nameauth@etoksd]}}%
1519     \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1520     \ex\ex\ex\ex\ex\ex\ex S\ex\ex\ex\ex\ex\ex\ex\@arga%
1521     \ex\ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex\ex{%
1522     \ex\ex\ex\ex\ex\ex\ex\ex\@nameauth@FirstNametrue%
1523     \ex\ex\ex\ex\ex\ex\ex\ex\NameauthFName\ex\ex\ex\ex\ex\ex\ex[%
1524     \ex\ex\ex\the\ex\ex\ex\@nameauth@etoksb\ex\ex\ex}%
1525     \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
1526     \the\@nameauth@etoksd]}}%
1527     \fi
1528     \fi
1529     \ignorespaces%
1530   }\ignorespaces%
1531 }\endgroup\ignorespaces}

```

4 Change History

0.7	General: Initial release	1	2.0	\@nameauth@Root: Trim spaces	69
0.75	\ForgetName: Add new argument . . .	96	\@nameauth@Actual: Added	79	
	\IndexName: Has current args	84	\@nameauth@Index: New tagging	78	
0.85	\@nameauth@Name: Hide commas	71	\@nameauth@Name: Trim spaces; redesign tagging	71	
	\AKA: Hide commas	99	\AKA: Trim spaces; fix tagging	99	
	\IndexName: Hide commas	84	\IndexActual: Added	82	
0.9	\@nameauth@Root: Expands	69	\IndexName: Fix spaces, tagging	84	
	\@nameauth@Suffix: Added	69	\PretagName: Added	91	
	\@nameauth@Suffix: Added	69	\TagName: Redesign tagging	92	
	\AKA*: Added	100	\UntagName: Redesign tagging	92	
	\FName: Added	84	General: Use dtxgen template; prevent malformed input	1	
	\SubvertName: Added	98	nameauth: Better arg handling	100	
0.94	\@nameauth@Index: Added	78	2.1	\@nameauth@Name: Fix Unicode	71
	\CapThis: Added	79		\AKA: Fix Unicode	99
	\ExcludeName: Added	88		\AccentCapThis: Added	79
	\IndexActive: Added	82	2.11	nameauth: Bugfix	100
	\IndexInactive: Added	82	2.2		
1.2	\TagName: Added	92		\NameauthFName: Added	67
	\UntagName: Added	92		\NameauthName: Added	67
1.26	\AKA: Fix name suffixes	99	2.3	\@nameauth@Name: Now internal	71
	\IndexName: Affixes now correct	84		\AKA: Expand starred mode	99
1.4	\@nameauth@Root: More robust	69		\ExcludeName: New xref test	88
	\ShowComma: Added	81		\FName: Interface macro	84
1.5	\@nameauth@Suffix: Trim spaces	69		\FName*: Interface macro	84
	\@nameauth@Name: Reversing/caps	71		\ForgetName: Global or local	96
	\AKA: Reversing and caps	99		\GlobalNames: Added	82
	\AllCapsActive: Added	79		\IfAKA: Added	95
	\AllCapsInactive: Added	79		\IfFrontName: Added	95
	\CapName: Added	79		\IfMainName: Added	95
	\RevComma: Added	80		\LocalNames: Added	82
	\RevName: Added	79		\Name: Interface macro	84
	\ReverseActive: Added	79		\Name*: Interface macro	84
	\ReverseCommaActive: Added	80		\NameauthLName: Added	67
	\ReverseCommaInactive: Added	80	2.4	\PName: Work directly with hooks	100
	\ReverseInactive: Added	79		\SubvertName: Global or local	98
1.6	nameauth: Environment added	100		\@nameauth@Hook: Current form	77
1.9	\ForgetName: Ensure global undef	96		\@nameauth@Name: Set token regs	71
	\KeepAffix: Added	81		\FrontNameHook: Added	67
	\TagName: Fix cs collisions	92		\GlobalNames: Ensure global	82
	\UntagName: Ensure global undef, fix cs collisions	92		\IfAKA: Test for excluded	95
	nameauth: Bugfix	100		\LocalNames: Ensure global	82
				\MainNameHook: Added	67
				\NameAddInfo: Added	93
				\NameClearInfo: Added	94
				\NameQueryInfo: Added	94

2.41		\@nameauth@Name: Enhanced workflow control	71
	\@nameauth@Name: Fix token regs . . .		71
	\AKA: Fix token regs		99
	nameauth: No local \newtoks		100
2.5		\@nameauth@Hook: Improve hooks . . .	77
	\@nameauth@Name: Current parsing approach		71
	\FrontNamesFormat: Added		67
	General: No default formatting		1
2.6		\@nameauth@Name: Better indexing . .	71
	\AKA: Fix index commas		99
	\IndexName: Fix commas		84
	\NoComma: Added		81
	General: Fix older syntax		1
3.0		\@nameauth@@Root: Redesigned	69
	\@nameauth@@Suffix: New test		69
	\@nameauth@@TrimTag: Added		69
	\@nameauth@Error: Added		71
	\@nameauth@Hook: Fix punct. detection		77
	\@nameauth@Name: Redesigned		71
	\@nameauth@NonWest: Added		75
	\@nameauth@Parse: Added		73
	\@nameauth@TrimTag: Added		69
	\@nameauth@UTFtest: Added		69
	\@nameauth@West: Added		76
	\AKA: Redesigned		99
	\DropAffix: Added		81
	\ExcludeName: Redesigned		88
	\ForceFN: Added		80
	\IfAKA: Redesigned		95
	\IncludeName: Added		89
	\IncludeName*: Added		90
	\IndexName: Redesigned		84
	\IndexRef: Added		86
	\NameParser: Added		82
	\SeeAlso: Added		82
3.01		\@nameauth@Error: Fixed	71
3.02		\@nameauth@NonWest: Restrict \ForceFN	75
3.03		\NameParser: First name only with “short” macros	82
3.1		\@nameauth@CC@p: Added	70
	\@nameauth@CC@pUTF: Added		70
	\@nameauth@Cap: Added; old caps gone		70
	\@nameauth@Name: Enhanced workflow control		71
	\@nameauth@Parse: Enhanced, integrated caps		73
	\@nameauth@UTFtest: Override bypasses test		69
	\AKA: Can skip index		99
	\AltCaps: Added		81
	\AltFormatActive: Added		80
	\AltFormatActive*: Added		80
	\AltFormatInactive: Added		80
	\AltOff: Added		80
	\AltOn: Added		80
	\ForceName: Added		82
	\ForgetThis: Added		81
	\IncludeName*: Fixed		90
	\IndexName: Better tests		84
	\IndexRef: Better tests		86
	\JustIndex: Added		82
	\KeepName: Added		81
	\NameParser: Older syntax fixed; NBSP added		82
	\NameQueryInfo: Short macro		94
	\PName: Can skip index		100
	\SkipIndex: Added		82
	\SubvertName: Fix old syntax		98
	\SubvertThis: Added		82
	\textBF: Added		81
	\textIT: Added		81
	\textSC: Added		81
	\textUC: Added		81
	General: Fix logic, arg tests		1
3.2		\@nameauth@@GetSuff: Added	69
	\@nameauth@@Root: Renamed		69
	\@nameauth@@Suffix: Renamed		69
	\@nameauth@CC@p: Renamed		70
	\@nameauth@Cap: Non-UTF		70
	\@nameauth@CapUTF: Added		70
	\@nameauth@GetSuff: Added		69
	\@nameauth@Parse: Fix alt. format and Western affixes		73
	\@nameauth@TestToks: Added		69
	\@nameauth@TrimTag: Renamed		69
	\@nameauth@UTFtest: Non-suffix only		69
	\@nameauth@UTFtestS: Added		70
	\NameParser: Fix alt. format and Western affixes		82
	General: Use \MakeUppercase instead of \uppercase		1

5 Index

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

Symbols		
<code>\@nameauth@@GetSuff</code> . . .	99	Arouet, François-Marie <i>see</i> Voltaire
<code>\@nameauth@@Root</code>	92	Atatürk <i>see</i> Kemal, Mustafa
<code>\@nameauth@@Suffix</code> . . .	96	Attila the Hun 7 , 33
<code>\@nameauth@@TrimTag</code> . . .	94	
<code>\@nameauth@Actual</code>	530	B
<code>\@nameauth@C@p</code>	146	Babbage , Charles 28
<code>\@nameauth@C@pUTF</code>	149	Bernard of Clairvaux 46
<code>\@nameauth@Cap</code>	145	Bess, Good Queen <i>see</i> Elizabeth I
<code>\@nameauth@CapUTF</code>	148	Boëthius 24
<code>\@nameauth@CheckDot</code> . . .	163	C
<code>\@nameauth@Clean</code>	89	<code>\CapName</code> 20 , 534
<code>\@nameauth@Error</code>	170	<code>\CapThis</code> 21 , 531
<code>\@nameauth@EvalDot</code>	165	Carnap, Rudolph 25 , 38 , 41
<code>\@nameauth@GetSuff</code>	98	Carter, J.E., Jr., pres. 14 , 42
<code>\@nameauth@Hook</code>	448	Carter, Jimmy <i>see</i> Carter, J.E., Jr.
<code>\@nameauth@Index</code>	496	Chaplin, Charlie 48
<code>\@nameauth@Name</code>	183	Chiang Kai-shek†, pres. 9 , 21
<code>\@nameauth@NonWest</code>	348	Cicero, M.T. 16 , 17 , 26
<code>\@nameauth@Parse</code>	230	Clemens, Samuel L. <i>see</i> Twain, Mark
<code>\@nameauth@Root</code>	91	Colfax, Schuyler, v.p. . . . 37
<code>\@nameauth@Suffix</code>	95	Confucius . 16 , 17 , 21 , 26 , 39
<code>\@nameauth@TestDot</code>	151	cummings, e.e. 22
<code>\@nameauth@TestToks</code>	100	D
<code>\@nameauth@TrimTag</code>	93	Dagobert I†, king 9
<code>\@nameauth@UTFtest</code>	108	DAVIS, Sammy, JR. 56 , 57 , 61
<code>\@nameauth@UTFtestS</code>	125	<code>de Smet</code> , Pierre-Jean 58 , 61
<code>\@nameauth@West</code>	400	de Soto, Hernando 7 , 21
<code>\@nameauth@toksa</code>	53	Demetrius I Soter, king . . 31
<code>\@nameauth@toksb</code>	53	<i>Doctor angelicus</i> <i>see</i> Thomas Aquinas
<code>\@nameauth@toksc</code>	53	<i>Doctor mellifluus</i> <i>see</i> Bernard of Clairvaux
A		Dongen, Marc van 2 , 71
<code>\AccentCapThis</code>	22 , 532	<code>\DropAffix</code> 18 , 585
ADAMS, John, pres. 56 , 57 , 61		Du Bois, W.E.B. 47
Æthelred II, king 8 , 21 , 24 , 33		du Cange <i>see</i> du Fresne, Charles
<code>\AKA</code> 42 , 1392		du Fresne, Charles 43
<code>\AKA*</code> 42 , 1438		DuBois, W.E.B. <i>see</i> Du Bois, W.E.B.
<code>\AllCapsActive</code>	20 , 536	E
<code>\AllCapsInactive</code>	20 , 535	Einstein, Albert 16 , 17 , 26 , 41
<code>\AltCaps</code> 29 , 569		
<code>\AltFormatActive</code>	27 , 547	
<code>\AltFormatActive*</code>	27 , 551	
<code>\AltFormatInactive</code>	27 , 555	
<code>\AltOff</code> 29 , 564		
<code>\AltOn</code> 29 , 559		
Anthony, Susan B. 41		
Arai Akino 20		
Aristotle 6 , 7		
		Elizabeth I, queen 3 , 6 , 7 , 16 , 17 , 26 , 38 , 43 , 44
		environments: <code>nameauth</code> 6 , 1449
		<code>\ExcludeName</code> 32 , 853
		F
		<code>\FName</code> 17 , 687
		<code>\FName*</code> 17 , 688
		<code>\ForceFN</code> 17 , 540
		<code>\ForceName</code> 25 , 592
		<code>\ForgetName</code> 41 , 1274
		<code>\ForgetThis</code> 41 , 590
		Friedrich I Barbarossa, em- peror 23
		<code>\FrontNameHook</code> 25 , 51
		<code>\FrontNamesFormat</code> 25 , 50
		FUKUYAMA Takeshi 28
		G
		GARBO, Greta 28
		<code>\GlobalNames</code> 41 , 594
		Goethe, J.W. von 8 , 18 , 21 , 31
		Gossett, Louis, Jr. 18
		Grant, Ulysses S., pres. . . 37
		Gregorio, Enrico 2
		Gregory I, pope 35 , 43
		Gregory the Great <i>see</i> Gregory I
		H
		Hammerstein, Oskar, II 18 , 21
		Harnack, Adolf 31
		Harun AL-RASHID 56 , 57 , 61
		Hearn, Lafcadio 43
		Henry VIII†, king 9
		Hope, Bob 38 , 43
		Hope, Leslie Townes <i>see</i> Hope, Bob
		I
		<code>\if@nameauth@InAKA</code> 53
		<code>\if@nameauth@InName</code> 53
		<code>\IfAKA</code> 39 , 1239
		<code>\IfFrontName</code> 38 , 1216
		<code>\IfMainName</code> 38 , 1193
		<code>\IncludeName</code> 33 , 945
		<code>\IncludeName*</code> 33 , 980
		<code>\IndexActive</code> 30 , 598

<code>\IndexActual</code>	34, 599		
<code>\IndexInactive</code>	30, 595		
<code>\IndexName</code>	31, 690		
<code>\IndexRef</code>	32, 752		
Iron Mike	<i>see</i> Tyson, Mike		
Ishida Yoko†	20		
J			
Janos, James			
.	<i>see</i> Ventura, Jesse		
<code>\JustIndex</code>	30, 597		
K			
Kanno, Yoko†	20		
<code>\KeepAffix</code>	18, 586		
<code>\KeepName</code>	18, 587		
Kemal, Mustafa	54		
Keynes, John Maynard	26		
King, Martin Luther, Jr.	24		
Koizumi Yakumo			
.	<i>see</i> Hearn, Lafcadio		
Konoe, Fumimaro†, PM	5, 7, 19, 21		
Kresge, Joseph	<i>see</i>		
.	Kreskin, The Amazing		
Kreskin, The Amazing	42		
L			
Lao-tzu	43		
Leo I, pope	35		
Leo the Great	<i>see</i> Leo I		
Lewis, Clive Staples	4, 7, 8, 17		
Li Er	<i>see</i> Lao-tzu		
<code>\LocalNames</code>	41, 593		
Louis XIV, king	18, 32, 43		
Lovelace, Ada	28		
Lueck, Uwe	2, 71		
Luecking, Dan	49		
Łukasiewicz, Jan	33		
LUTHER, Martin	29		
M			
Maimonides			
.	45, <i>see also</i> Rambam		
<code>\MainNameHook</code>	25, 49		
Malebranche, Nicolas	25		
MEDICI, Catherine de'	21, 29		
MENCIUS	56, 57, 61		
MENG Ke	<i>see</i> MENCIUS		
Miyazaki Hayao	8, 16, 17, 31		
Molnár, Freneč†	5, 19		
Moses ben-Maimon			
.	<i>see</i> Maimonides		
N			
<code>\Name</code>	16, 684		
<code>\Name*</code>	16, 685		
<code>\NameAddInfo</code>	36, 1124		
nameauth (environment)	6, 1449		
<code>\NameauthFName</code>	54, 62		
<code>\NameauthLName</code>	53, 62		
<code>\NameauthName</code>	52, 62		
<code>\NameClearInfo</code>	37, 1170		
<code>\NameParser</code>	58, 602		
<code>\NameQueryInfo</code>	37, 1147		
<code>\NamesActive</code>	25, 589		
<code>\NamesFormat</code>	25, 48		
<code>\NamesInactive</code>	25, 588		
<code>\NoComma</code>	18, 584		
O			
Oberdiek, Heiko	2, 68		
P			
Patton, George S., Jr.	4, 7		
<code>\PName</code>	46, 1439		
<code>\PName*</code>	46, 1448		
<code>\PretagName</code>	33, 1003		
R			
Rambam	45,		
.	<i>see also</i> Maimonides		
<code>\RevComma</code>	21, 541		
<code>\ReverseActive</code>	19, 539		
<code>\ReverseCommaActive</code>	21, 545		
<code>\ReverseCommaInactive</code>	21, 543		
<code>\ReverseInactive</code>	19, 538		
<code>\RevName</code>	19, 537		
Rockefeller, Jay <i>see</i> Rockefeller, John David, IV			
ROCKEFELLER, John David, III	28		
Rockefeller, John David, IV	4, 7, 8		
RÜHMANN, Heinrich Wilhelm			
.	<i>see</i> RÜHMANN, Heinz		
RÜHMANN, Heinz	44		
S			
Schlicht, Robert	2		
<code>\SeeAlso</code>	32, 601		
SHAKESPEARE, Wm.	59		
<code>\ShowComma</code>	18, 583		
<code>\SkipIndex</code>	30, 596		
Smith, John*	36		
Smith, John* (second)	36		
Smith, John* (third)	36		
Snel van Royen, R.	45		
Snel van Royen, W.	45		
Snellius			
.	<i>see</i> Snel van Royen, R.; Snel van Royen, W.		
Stephani, Philipp	2		
Strietelmeier, John	20		
<code>\SubvertName</code>	41, 1333		
<code>\SubvertThis</code>	41, 591		
Sullenberger, Chesley B., III	17, 31		
Sun King	<i>see</i> Louis XIV		
Sun Yat-sen, pres.	3, 18		
T			
<code>\TagName</code>	35, 1050		
<code>\textBF</code>	27, 581		
<code>\textIT</code>	27, 579		
<code>\textSC</code>	27, 575		
<code>\textUC</code>	27, 577		
Thomas à Kempis	23		
Thomas Aquinas	45		
TOKUGAWA Ieyasu	28		
Twain, Mark	46		
Tyson, Mike	46		
U			
<code>\UntagName</code>	36, 1101		
V			
Van Buren, Martin, pres.	8, 21		
Ventura, Jesse	39		
Vlad II Dracul	52		
Vlad III Dracula	52		
Vlad Țepeș			
.	<i>see</i> Vlad III Dracula		
Voltaire	46		
W			
Washington, George, pres.	3, 4, 6, 7, 30, 36, 52, 54		
White, E. B.	20, 64		
William I	46		
William the Conqueror			
.	<i>see</i> William I		
Y			
Yamamoto Isoroku	5, 7, 19		
Yohko	20		
Yoshida Shigeru†, PM	9		