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

Charles P. Schaum†

Released 2020/02/20

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**
1.1 How to Use the Manual . . . 2
1.2 Task Dashboard 3
1.3 Basic Concepts 4
1.3.1 Traditional Interface . 5
1.3.2 Simplified Interface . . 7
1.3.3 Macro Overview . . . 10
1.4 Obsolete Syntax 11
1.5 Name Pattern Overview . . . 11
1.6 Debug and Avoid Errors . . . 13

2 Detailed Usage **17**
2.1 Package Options 17
2.2 Naming Macros 21
2.2.1 \Name and \Name* . . 21
2.2.2 Forenames: \FName . . 22
2.2.3 Variant Names 23
2.3 Language Topics 25
2.3.1 Affixes 25
2.3.2 Listing by Surname . 26
2.3.3 Eastern Names 26
2.3.4 Particles / Ancient . . 27
2.4 Basic Formatting 32
2.5 Alternate Formatting 35
2.5.1 Basic Features 35
2.5.2 Advanced Features . . 37
2.6 Indexing Macros 39
2.6.1 Entries & Control . . 39
2.6.2 Cross-References . . . 40
2.6.3 Index Sorting 44
2.6.4 Index Tags 45
2.7 "Text Tags" 48
2.8 Name Decisions 49
2.8.1 Making Decisions . . . 50
2.8.2 Testing Decisions . . . 51
2.9 Alternate Name Macros . . . 54
2.10 Longer Examples 57
2.10.1 Hooks: Intro 57
2.10.2 Hooks: Life Dates . . 58
2.10.3 Hooks: Advanced . . . 60
2.10.4 Customization 67
2.11 Technical Notes 69
2.11.1 General 69
2.11.2 Active Unicode 70
2.11.3 LaTeX Engines 72

3 Implementation **74**
3.1 Flags and Registers 74
3.2 Hooks 76
3.3 Package Options 77
3.4 Internal Macros 78
3.5 Prefix Macros 91
3.6 General User Interface 94

4 Change History **115**

5 Index **117**

*This file describes version 3.3, last revised 2020/02/20.

†E-mail: charles dot schaum@comcast.net

1 Quick Start

1.1 How to Use the Manual

A **name authority** is a canonical, scholarly list of names to which all variants must refer. The task dashboard (Section 1.2) guides one to various areas of interest. Start with the basics and add features as needed. To load the defaults, simply type:

```
\usepackage{nameauth}
```

Package Design and Features

With nameauth names become abstractions: verbs that alter state and nouns that have state. That improves accuracy and consistency:

- **Automate** name forms used in professional writing. First uses of names will have full forms. Later uses have shorter forms. Names vary in the text, but stay constant in the index.
- Permit **complex name formatting**.
- Many **cross-cultural, multilingual naming conventions** are possible. More details appear in Sections 1.6, 2.5, 2.6.3, and 2.10.3.
- **Automatic sort keys and tags** aid indexing.
- One can **associate information with names.**
- The standard used for implementing the nameauth indexing macros is Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). All references [Mulvany] refer to this edition.
- In Section 1.6 we see how to avoid common errors.
- Section 2.11 contains **thanks** and various technical notes.

Special Signs

As teaching aids, this manual uses markings that are not part of nameauth, but in some cases are implemented using it:

We show first uses and subsequent uses of names (Sections 2.4, 2.8.1).

† A dagger indicates "non-native" Eastern forms (Section 2.3.3).

‡ A double dagger shows usage of the obsolete syntax (Section 1.4).

§ A section mark denotes index entries of fictional names.

3.0 ← Major changes have package version numbers in the margin.

 ← The "dangerous bend" shows where caution is needed.

Disclaimer

Names are about real people. This manual mentions notable figures both living and deceased. All names herein are meant to be used respectfully, for teaching purposes only. At no time is any disrespect or bias intended.

1.2 Task Dashboard

Here we link to sections by task in order to get things done quickly. Many sections have return links at their end that bring the reader back to this page.

Where do you want to go today?

Quick Start

Sections 1.3, 1.3.1, 1.3.2, 1.3.3, 1.4, 1.5, 1.6

Basics

Package options: Section 2.1

Basics

Name macros: Sections 2.2.1, 2.2.2

Intermediate

Variant forms: Sections 2.2.3, 2.3.4, 2.6.2

Intermediate

Avoid errors: Sections 1.4, 1.5, 1.6, 2.3.4, 2.11.2

Language

Western names: Sections 2.3.1, 2.3.2

Language

Eastern names: Sections 2.3.1, 2.3.3

Language

Particles, medieval, ancient names: Sections 2.3.4, 2.5, 2.10.3

Index

Index entries and control: Section 2.6.1

Index

Index cross-refs, automatic sorting, and auto-info: Sections 2.6.2, 2.6.3, 2.6.4, 2.9

Advanced

Generally manage how names are typeset: Sections 2.4, 2.5, 2.8.1, 2.8.2, 2.10.1 – 2.10.4

Advanced

Generally manage names by using a name authority: Sections 2.2.3, 2.4, 2.6.1, 2.8.1

Advanced

Make complex elements determined automatically by names: Sections 2.7, 2.8.2

Application

Use nameauth with beamer overlays to get correct name forms: Sections 2.4, 2.8, 2.8.1, 2.8.2

Application

History / game books, other complex layouts: Sections 2.4, 2.6.4, 2.7, 2.8.1, 2.8.2, 2.10.1 – 2.10.3

1.3 Basic Concepts

We encode names in macro arguments to address multiple naming systems. Required name elements are shown in **black**; optional parts are in **red**.[1] The arguments appear in the order ⟨*FNN*⟩ ⟨*SNN*⟩ ⟨*Affix*⟩ ⟨*Alternate*⟩:

Western Name

Forename(s): ⟨*FNN*⟩	**Surname(s): ⟨*SNN*⟩**	**Descriptor: ⟨*Affix*⟩**
Personal name(s): *baptismal name Christian name multiple names praenomen*[2]	Family name: *of father, mother ancestor, vocation origin, region nomen, cognomen patronym*	Sobriquet / title: *Sr., Jr., III. . . notable attribute origin, region*

Alternate Name(s): ⟨*Alternate*⟩

In the body text, not the index, ⟨*Alternate*⟩ swaps with ⟨*FNN*⟩ for Western names and ⟨*Affix*⟩ for all other name categories.

Eastern Name

Family name: ⟨*SNN*⟩	**Personal name: ⟨*Affix*⟩**	**Descriptor: ⟨*Alternate*⟩**
Family / clan name	Seldom multiple names; multi-character okay.	Title, etc. (old syntax for personal names)

Ancient name

Personal name: ⟨*SNN*⟩	**Descriptor: ⟨*Affix*⟩**	**Descriptor: ⟨*Alternate*⟩**
Given name(s)	Sobriquet / title: *Sr., Jr., III. . . notable attribute origin, region patronym*	Alternate name (old syntax for titles, etc.)

[1] Compare [Mulvany, 152–82] and the *Chicago Manual of Style*.

[2] How one handles Roman names depends on index entry form; some possible suggestions are given above. Explained on page 29 and following, we have a name Publius Cornelius Scipio that can be Scipio Africanus or just Scipio, using macro expansion.

1.3.1 Traditional Interface

Mandatory arguments are shown in **black**, with optional elements in **red**. If the required argument ⟨*SNN*⟩ expands to the empty string, nameauth will generate a package error. Extra spaces around each argument are stripped (Section 1.6). The argument patterns shown here are used in many nameauth macros.

Western Names

	Required forename(s)	Required surname(s), optional ⟨*Affix*⟩	Optional, in text only

`\Name`
`\Name*`
`\FName`
	`[`⟨***FNN***⟩`]`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

Examples

Western names require the ⟨*FNN*⟩ argument to be present. One always includes all arguments for consistent index entries. The simplified interface (Section 1.3.2) cuts down the amount of typing in many cases.

`\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.

The ⟨*Alternate*⟩ argument will swap with ⟨*FNN*⟩ in the text, not in the index or the name pattern (Section 1.5). To see alternate names, one must use a macro that shows forenames (first use, `\Name*`, and `\FName`). Western names require a comma to delimit affixes (Section 2.3.1). Below we see alternate names:

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

`\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

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

In addition to alternate forenames, one also can display alternate surnames, but that uses several different approaches (Sections 2.2.3, 2.3.4, 2.5.2, 2.10.3).

"Non-native" Eastern Names, Western Index Entry

	Required forename(s)	Required surname(s), no ⟨*Affix*⟩	Optional, in text only
`\Name` `\Name*` `\FName`	`[`⟨***FNN***⟩`]`	`{`⟨***SNN***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN***⟩`}` if other text in brackets `[]` follows.

Examples

Below we start with "regular" Western name forms:

`\Name[Hideyo]{Noguchi}` . . . . . Hideyo Noguchi
`\Name*[Hideyo]{Noguchi}[Doctor]` . . . . . Doctor Noguchi
`\Name[Frenec]{Molnár}` . . . . . Frenec Molnár

To turn them into "non-native" Eastern names or proper Hungarian names [Mulvany, 166] we use the reversing macros and leave the ⟨*Alternate*⟩ argument empty (Section 2.3.3). Index entries are in Western style: ⟨*SNN*⟩, ⟨*FNN*⟩:

`\CapName\RevName\Name*[Hideyo]{Noguchi}` . . . . . NOGUCHI Hideyo†
`\CapName\RevName\Name*[Hideyo]{Noguchi}[Sensei]` . . . . NOGUCHI Sensei†
`\RevName\Name*[Frenec]{Molnár}` . . . . . Molnár Frenec†

Reversed Western forms do not work with the older syntax (Section 1.4) and **they do not share name control sequences and index entries** with "native" Eastern names and ancient name forms (Section 1.5).

"Native" Eastern Names in the Text, Eastern Index Entry

	Required surname and forename	Optional, in text only
`\Name` `\Name*` `\FName`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

Examples

The comma-delimited required argument, ⟨*SNN, Affix*⟩, is the key to non-Western names, which always take the form ⟨*SNN Affix*⟩ in the index. See Section 2.3.3. "Native" Eastern names have Eastern name order from the start and **they do not share name control sequences and index entries** with Western names (Section 1.5). They can be reversed to have Western name order in the body text.

3.0 Except for mononyms, non-Western forms also can have alternate names. This is incompatible with the older syntax (see Section 1.4). Unless the index must have Western-style entries, "native" forms are best for Eastern names:

```
\Name{Yamamoto, Isoroku} ..... Yamamoto Isoroku
\Name{Yamamoto, Isoroku} ..... Yamamoto
\RevName\Name*{Yamamoto, Isoroku}[Admiral] ..... Admiral Yamamoto
\Name{Miyazaki, Hayao} ..... Miyazaki Hayao
\Name*{Miyazaki, Hayao}[Sensei] ..... Miyazaki Sensei
\RevName\Name*{Miyazaki, Hayao}[Mr.] ..... Mr. Miyazaki
```

Ancient and Medieval Names

	Required name optional ⟨*Affix*⟩	Optional, in text only
`\Name` `\Name*` `\FName`	`{`⟨***SNN, Affix***⟩`}`	`[`⟨***Alternate***⟩`]`

Add braces `{}` after `{`⟨***SNN, Affix***⟩`}` if other text in brackets `[]` follows.

Examples

These forms are meant for royalty and ancient figures. They can be mononyms or have multiple names, and may have an affix. Note the teaser for Section 2.2.2:

```
\Name{Aristotle} ..... Aristotle
\Name{Aristotle} ..... Aristotle
\Name{Elizabeth, I} ..... Elizabeth I
\Name{Elizabeth, I} ..... Elizabeth
\ForceFN\FName{Elizabeth, I}[Good Queen Bess] ..... Good Queen Bess
```

1.3.2 Simplified Interface

`nameauth` Although not required, using the `nameauth` environment in the preamble guards against undefined macros. This environment defines a tabular-like macro:

```
\begin{nameauth}
    \< ⟨arg1⟩ & ⟨arg2⟩ & ⟨arg3⟩ & ⟨arg4⟩ >
\end{nameauth}
```

It uses ⟨*arg1*⟩ as a basis to create three macros that are equivalent to:

```
\⟨arg1⟩  → \Name[⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]
\L⟨arg1⟩ → \Name*[⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]   % L for long
\S⟨arg1⟩ → \FName[⟨arg2⟩]{⟨arg3⟩}[⟨arg4⟩]   % S for short
```

If either ⟨*arg1*⟩ or ⟨*arg3*⟩ are empty, or ⟨*SNN*⟩ is empty, nameauth will generate a package error. Forgetting the backslash, ampersands, or angle brackets will cause errors. For more on ⟨*arg4*⟩ see page 9.

Comments below are not part of the environment. Extra spaces in each argument are stripped (Section 1.6). Put trailing braces `{}` or something else after the shorthand macros if text in brackets `[]` follows, so it does not become an optional argument. Below we introduce name forms with particles.

Examples

```
\begin{nameauth}
%    ⟨arg1⟩    ⟨arg2⟩           ⟨arg3⟩               ⟨arg4⟩
  \< Wash    & George         & Washington        &        >% Western
  \< Harnack & Adolf          & Harnack           &        >% Western
  \< Lewis   & Clive Staples & Lewis             &        >% Western
  \< Pat     & George S.      & Patton, Jr.       &        >% W. affix
  \< JRIV    & John David     & Rockefeller, IV   &        >% W. affix
  \< Ches    & Chesley B.     & Sullenberger, III &        >% W. affix
  \< Soto    & Hernando       & de Soto           &        >% W. part.
  \< JWG     & J.W. von       & Goethe            &        >% W. part.
  \< VBuren  & Martin         & Van Buren         &        >% W. part.
  \< Noguchi & Hideyo         & Noguchi           &        >% W. as E.
  \< Miyaz   &                & Miyazaki, Hayao   &        >% Eastern
  \< Yamt    &                & Yamamoto, Isoroku &        >% Eastern
  \< Aeth    &                & Æthelred, II      &        >% Ancient
  \< Attil   &                & Attila, the Hun   &        >% Ancient
  \< Dem     &                & Demetrius, I      &        >% Ancient
  \< Eliz    &                & Elizabeth, I      &        >% Ancient
  \< Aris    &                & Aristotle         &        >% Mono
  \< CSL     & Clive Staples & Lewis             & C.S.   >% W. alt.
  \< MSens   &                & Miyazaki, Hayao   & Sensei >% E. alt.
\end{nameauth}
```

Output	Short Form	Long Form
Washington	`\Wash`	`\Name[George]{Washington}`
George Washington	`\LWash`	`\Name*[George]{Washington}`
George	`\SWash`	`\FName[George]{Washington}`
	`\JustIndex\Wash`	`\IndexName[George]{Washington}`
Elizabeth	`\SubvertThis\Eliz`	`\SubvertThis\Name{Elizabeth,I}`
Elizabeth I	`\ForgetThis\Eliz`	`\ForgetThis\Name{Elizabeth,I}`

English and modern Romance languages keep the particle with the surname. German and other languages do not (cf. Sections 2.3.4 and 2.5).[3]

Macro	Body Text	Index
`\VBuren`	Martin Van Buren	Van Buren, Martin
`\VBuren`	Van Buren	Van Buren, Martin
`\Soto`	Hernando de Soto	de Soto, Hernando
`\CapThis\Soto`	De Soto	de Soto, Hernando
`\JWG`	J.W. von Goethe	Goethe, J.W. von
`\JWG`	Goethe	Goethe, J.W. von

[3] See also [Mulvany, 152–82], and the *Chicago Manual of Style*.

⟨*Alternate*⟩ Tips

Above we listed two shorthands that had alternate names in ⟨*arg4*⟩: `\CSL` and `\MSens`. They have related shorthands whose ⟨*arg4*⟩ are empty: `\Lewis` and `\Miyaz`. Here are how they are related (cf. Section 2.2.2):

- They share the same name patterns (Section 1.5): `\ForgetThis\Lewis` Clive Staples Lewis; `\CSL` Lewis. `\ForgetThis\Miyaz` Miyazaki Hayao; and `\MSens` Miyazaki.
- More commonly, one produces alternate names with `\LLewis[C.S.]` C.S. Lewis and `\LMiyaz[Sensei]` Miyazaki Sensei.

- Both `\LCSL` C.S. Lewis and `\LMSens` Miyazaki Sensei already have ⟨*Alternate*⟩ built in. They cannot take another optional argument. One must remember which shorthands have used ⟨*arg4*⟩.
- If one should forget that, one will get errors: `\LCSL[C.S.]` C.S. Lewis[C.S.] and `\LMsens[Sensei]` Miyazaki Sensei[Sensei].

Variant Overview

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

Particles: (Section 2.3.4)
`\Soto` . . . Hernando de Soto
`\Soto` . . . de Soto
`\CapThis\Soto` . . . De Soto

Affixes: (Section 2.3.1)
`\Pat` . . . George S. Patton Jr.
`\LPat` . . . George S. Patton Jr.
`\DropAffix\LPat` . . . George S. Patton
`\Pat` . . . Patton
`\SPat` . . . George S.

Nicknames: (Section 2.2.2)
`\JRIV` . . . John David Rockefeller IV
`\DropAffix\LJRIV[Jay]` . . Jay Rockefeller
`\SJRIV[Jay]` . . . Jay
`\Lewis` . . . Clive Staples Lewis
`\LLewis[Jack]` . . . Jack Lewis
`\SLewis[Jack]` . . . Jack
`\LCSL` . . . C.S. Lewis
`\SCSL` . . . C.S.

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

Ancient / Royal: (Sections 2.3.3, 2.3.4)
`\Aeth` . . . Æthelred II
`\Aeth` . . . Æthelred
`\LAeth[Unrædig]` . . . Æthelred Unrædig
`\Attil` . . . Attila the Hun
`\Attil` . . . Attila

"Non-native" Eastern: (Section 2.3.3)
`\Noguchi` . . . Hideyo Noguchi
`\LNoguchi` . . . Hideyo Noguchi
`\LNoguchi[Doctor]` . . . Doctor Noguchi
`\SNoguchi` . . . Hideyo
`\RevName\LNoguchi` . . . Noguchi Hideyo†
`\CapName\RevName\LNoguchi` . NOGUCHI Hideyo†
`\CapName\Noguchi` . . . NOGUCHI†

"Native" Eastern: (Section 2.3.3)
`\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

Above we used `\ForgetThis` (Section 2.8.1) to reset first uses of names. Now we set up examples on page 41 by invoking `\ExcludeName{Attila, the Hun}` and `\AKA[John David]{Rockefeller,IV}[Jay]{Rockefeller}` Jay Rockefeller. On why that form has a different index entry than `\DropAffix\LJRIV[Jay]` Jay Rockefeller, see Sections 2.6.2 and 2.9.

1.3.3 Select Macro Overview

Below we have a partial selection of macros and their arguments in overview:

⟨*prefix macros*⟩	\Name	⟨*optional **⟩	⟨*name args*⟩	
⟨*prefix macros*⟩	\FName	⟨*optional **⟩	⟨*name args*⟩	
⟨*prefix macros*⟩	\IndexName		⟨*name args*⟩	
⟨*prefix macros*⟩	\IndexRef		⟨*xref args*⟩	⟨*target*⟩
⟨*prefix macros*⟩	\AKA	⟨*optional **⟩	⟨*target args*⟩	⟨*xref args*⟩
	\ExcludeName		⟨*name args*⟩	
	\IncludeName	⟨*optional **⟩	⟨*name args*⟩	
	\PretagName		⟨*name args*⟩	⟨*sort key*⟩
	\TagName		⟨*name args*⟩	⟨*tag*⟩
	\UntagName		⟨*name args*⟩	
	\NameAddInfo		⟨*name args*⟩	⟨*tag*⟩
	\NameQueryInfo		⟨*name args*⟩	
	\NameClearInfo		⟨*name args*⟩	
	\ForgetName		⟨*name args*⟩	
	\SubvertName		⟨*name args*⟩	
	\IfMainName		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
	\IfFrontName		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}
	\IfAKA		⟨*name args*⟩	{⟨*y*⟩}{⟨*n*⟩}{⟨*x*⟩}

The ⟨*prefix macros*⟩ below have a one-time effect per name and they also stack. For example: `\CapThis\RevName\SkipIndex\Name[bar]{foo}`: Foo Bar.

	Capitalization in the Text
\CapThis	Capitalize first letter of all name components in body text.
\AccentCapThis	Fallback when Unicode detection cannot be done.
\CapName	Cap entire ⟨*SNN*⟩ in body text. Overrides \CapThis.
	Reversing in the Text
\RevName	Reverse order of any name in body text. Overrides \RevComma
\RevComma	Reverse only Western names to ⟨*SNN*⟩, ⟨*FNN*⟩.
	Commas in the Text
\ShowComma	Add comma between ⟨*SNN*⟩ and ⟨*Affix*⟩.
\NoComma	No comma between ⟨*SNN*⟩ and ⟨*Affix*⟩. Overrides \ShowComma.
	Name Breaks in the Text
\DropAffix	Drop affix only for a long Western name reference.
\KeepAffix	Insert non-breaking space (NBSP) between ⟨*SNN*⟩, ⟨*FNN/Affix*⟩.
\KeepName	Insert NBSP between all name elements. Overrides \KeepAffix.
	Forcing Name Forms in the Text
\ForgetThis	Force a first-time name use. Negates \SubvertThis.
\SubvertThis	Force a subsequent use.
\ForceName	Force first-use formatting hooks.
\ForceFN	Force printing of ⟨*Affix*⟩ in non-Western short forms.
	Indexing
\SeeAlso	Make a *see also* reference instead of a page reference. Only for use with \IndexRef, \AKA, \PName and their starred variants.
\SkipIndex	Do not create index entries.
\JustIndex	Act like \IndexName; negated by \AKA, \PName.

Back to Section 1.2

1.4 Obsolete Syntax

This "ghost" of nameauth past limits alternate names and cross-references (Section 2.9), excludes comma-delimited names, and complicates indexing and tagging (Sections 2.6.4 and 2.7). When the ⟨*FNN*⟩ and ⟨*Affix*⟩ arguments are empty, ⟨*Alternate*⟩ acts like ⟨*Affix*⟩ and affects both name and index patterns (Section 1.5). In this manual we designate these names with a double dagger (‡):

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

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

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

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

Henry VIII (*Defensor Fidei*)

Back to Section 1.2

1.5 Name Pattern Overview

The table below shows how the macro arguments generate name patterns central to nameauth. The ⟨*Alternate*⟩ argument only affects patterns when using the obsolete syntax. The naming macro arguments create internal control sequences that affect names in both the text and the index:

Macro Arguments	Patterns	Type
`[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}`	⟨*FNN*⟩!⟨*SNN*⟩	Western
`[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}`	⟨*FNN*⟩!⟨*SNN*⟩,⟨*Affix*⟩	Western
`{`⟨*SNN, Affix*⟩`}`	⟨*SNN*⟩,⟨*Affix*⟩	non-Western
`{`⟨*SNN*⟩`}[`⟨*Alt*⟩`]`	⟨*SNN*⟩,⟨*Alt*⟩	(obsolete)
`{`⟨*SNN*⟩`}`	⟨*SNN*⟩	mononym

We “forget” several names below to create first-use cases:

Macro	Body Text	\ShowPattern
\Harnack[Adolf von]	Adolf von Harnack	Adolf!Harnack
\LHarnack	Adolf Harnack	Adolf!Harnack
\ForgetThis\Pat	George S. Patton Jr.	GeorgeS.!Patton,Jr.
\DropAffix\LPat	George S. Patton	GeorgeS.!Patton,Jr.
\ForgetThis\Noguchi	Hideyo Noguchi	Hideyo!Noguchi
\RevName\LNoguchi	Noguchi Hideyo†	Hideyo!Noguchi
\ForgetThis\Yamt	Yamamoto Isoroku	Yamamoto,Isoroku
\RevName\LYamt	Isoroku Yamamoto	Yamamoto,Isoroku
\ForgetThis\Name{Henry,VIII}	Henry VIII	Henry,VIII
\Name*{Henry}[VIII]	Henry VIII‡	Henry,VIII
\Dem[I Soter]	Demetrius I Soter	Demetrius,I
\LDem	Demetrius I	Demetrius,I
\ForgetThis\Aris	Aristotle	Aristotle
\Aris	Aristotle	Aristotle

Six suffixes are appended to these patterns to create independent data sets:

Description	Pattern	Mnemonic	Example
Front-matter names	⟨*pattern*⟩!NF	“name front”	Adolf!Harnack!NF
Main-matter names	⟨*pattern*⟩!MN	“main name”	Hideyo!Noguchi!MN
Index cross-refs	⟨*pattern*⟩!PN	“pseudonym”	Yamamoto,Isoroku!PN
Index sorting tags	⟨*pattern*⟩!PRE	“pretag”	Henry,VIII!PRE
Index info tags	⟨*pattern*⟩!TAG	“tag”	Demetrius,I!TAG
“Text tag” database	⟨*pattern*⟩!DB	“database”	Aristotle!DB

The following macros **write** to these data sets; others also can read from them:

Macros	!NF	!MN	!PN	!PRE	!TAG	!DB
\Name \Name* \FName	■	■				
\ForgetName \SubvertName	■	■				
\PName\PName*	■	■	■			
\AKA \AKA* \IndexRef			■			
\ExcludeName			■			
\IncludeName \IncludeName*			■			
\PretagName				■		
\TagName \UntagName					■	
\NameAddInfo \NameClearInfo						■

Back to Section 1.2

1.6 Debugging and Avoiding Errors

Debugging Macros

\ShowPattern **3.3**

`\ShowPattern` displays how the name arguments create name patterns. One can debug pattern collisions and other issues with this macro:

`\ShowPattern[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

We used `\ShowPattern` in two of the tables on the previous page in order to illustrate name control patterns. We set the macro using a typewriter font, e.g.: `\texttt{\ShowPattern[Hernando]{de Soto}}`: `Hernando!deSoto`

\ShowIdxPageref
\ShowIdxPageref*
3.3

`\ShowIdxPageref` displays a full index entry in the text. Its analogue is `\ShowIdxPageref*`, which shows a short index entry. Both only show names as page references, even if they are cross-references:

`\ShowIdxPageref [`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`
`\ShowIdxPageref*[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

The full entry produced by `\ShowIdxPageref` can be affected by both index styles and tags produced by `\PretagName` and `\TagName`, e.g.:

`\texttt{\ShowIdxPageref[Hernando]{de Soto}}:`
Desoto, Hernando=de Soto, Hernando|hyperpage

`\ShowIdxPageref*` appears throughout this manual to illustrate basic index entries, e.g.: `\ShowIdxPageref*[Hernando]{de Soto}`: de Soto, Hernando

Avoiding Common Errors

- Keep it simple! Avoid unneeded macros and use the simplified interface.
- Compare index entries with names in the body text.
- Check package warnings. Set the `verbose` option if needed.
- Check arguments' braces and brackets to avoid errors like "`Paragraph ended`" and "`Missing` ⟨*grouping token*⟩ `inserted.`"
- Do not format ⟨*SNN*⟩,⟨*Affix*⟩ together as a pair. Format ⟨*SNN*⟩ and ⟨*Affix*⟩ separately (Section 2.5).
- Sort names in the index with `\PretagName` (Section 2.6.3).
- In package docs (`dtx` files) set up the `nameauth` environment and tags in the driver section to avoid errors.

Obsolete Syntax Caution

- The older syntax has restrictions (Section 1.4). Only the new syntax permits variant names, e.g.: `\Name*{Henry, VIII}[Tudor]` Henry Tudor. The new syntax is preferred.
- A proper form for the old syntax is `\Name*{Henry}[VIII]`: Henry VIII.
- `\Name[Henry]{VIII}` is a malformed Western name: "Henry VIII" and "VIII." Likewise `\Name[Henry]{VIII}[Tudor]`: "Tudor VIII" and "VIII." Both have the incorrect index entry "VIII, Henry".

Standard Warnings

- If one defines shorthand macros in the `nameauth` environment whose control sequence already exists, warnings always appear. For example:

```
\PretagName[E.\,B.]{White}{White, E.B.}
\begin{nameauth}
  \< White & E.B.   & White & > % v.1
  \< White & E.\,B. & White & > % v.2
\end{nameauth}
```

- `\White` gives "E. B. White". Its pattern is: `E.\,B.!White`. We lost the first version. We forget White for later. There should be two package warnings in this section for the redefinition of `\White`, because we defined it in the driver, then again here twice.
- This could be a problem if a name shorthand replaces an actual macro that is used for something else and breaks that macro.
- Then again, if one uses, e.g., a new `nameauth` environment per chapter, these warnings may be harmless. User discretion is advised.

Verbose Warnings

Package warnings result from the following **only** when the `verbose` option is used because we do not want the default to be "chatty":

- Creating an index page reference after using a name as an xref or excluding it. Not allowed.
- Creating the same cross-reference multiple times. Not allowed.
- Using `\ExcludeName` on an xref. Not allowed.

3.3

- Using `\IncludeName` on an xref. Not allowed.
- Using `\ExcludeName` to exclude a name that exists is allowed, but a warning still results.
- `\PretagName` sorts xrefs, but also creates "informational warnings."
- Using `\TagName` and `\UntagName` on xrefs. No tag allowed.

Error Protection Strategies

extra spaces

The nameauth package trims extra spaces **around** name arguments to prevent errors like multiple index entries that appear due to extra spaces. LaTeX usually compacts internal spaces. For example, instead of being two different names, below we have the same name in a first, then subsequent use:

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

This does not include explicit spaces from `\space`, etc. For example, the pattern `MartinLuther!King,Jr.` comes from `\Name[Martin Luther]{King, Jr.}` while `Martin~Luther!King,Jr.` comes from `\Name[Martin~Luther]{King, Jr.}`. The tilde appears as a non-breaking space in the text.

full stop detection

Full stops appear in one's initials and in affixes like "Jr." (junior), "Sr." (senior), "d. J." (*der Jüngere*), and "d. Ä." (*der Ältere*). The naming macros and some of the alternate name macros (Section 2.9) check if the printed name ends with a full stop and is followed by one. They gobble the extra full stop:

```
This is Rev. Dr. \Name[Martin Luther]{King, Jr.}.
```
This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

```
This is Rev. Dr. \Name[Martin Luther]{King, Jr.}.
```
This is Rev. Dr. King. Full stop is not gobbled.

```
Again we speak fully of \Name*[Martin Luther]{King, Jr.}.
```
Again we speak fully of Martin Luther King Jr. Full stop is gobbled.

```
We drop the affix: \DropAffix\Name*[Martin Luther]{King, Jr.}.
```
We drop the affix: Martin Luther King. Full stop is not gobbled.

```
His initials are \FName[Martin Luther]{King, Jr.}[M.L.].
```
His initials are M.L. Full stop is gobbled.

grouping issues

Take care when using braces and spaces with a name at the end of a sentence. Braces can change name arguments, even though they look the same. We disable indexing for the three points below:

- If one encapsulates a name in braces, the punctuation detection fails:
  ```
  This is Rev. Dr. {\Name*[Martin Luther]{King, Jr.}}.
  ```
  This is Rev. Dr. Martin Luther King Jr.. Full stop is not gobbled.

  A solution encapsulates both the name and the full stop:
  ```
  This is Rev. Dr. {\Name*[Martin Luther]{King, Jr.}.}
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.
- If one encapsulates ⟨*Affix*⟩ in braces, the punctuation detection fails:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, {Jr.}}.
  ```
  This is Rev. Dr. Martin Luther King Jr.. Full stop is not gobbled.

  The solution leaves the full stop in ⟨*Affix*⟩ outside the braces:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, {Jr}.}.
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

  The name patterns, however, are different, creating different names:
  ```
  MartinLuther!King,{Jr.}
  MartinLuther!King,{Jr}.
  ```
- If one leaves an extra space after a name, the punctuation detection fails:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, Jr.}␣.
  ```
  This is Rev. Dr. Martin Luther King Jr. . Full stop is not gobbled.

  The solution removes the extra space:
  ```
  This is Rev. Dr. \Name*[Martin Luther]{King, Jr.}.
  ```
  This is Rev. Dr. Martin Luther King Jr. Full stop is gobbled.

active chars and macros

Variations in the use of active characters and control sequences also change name arguments and index sorting (Section 2.6.3; cf. 2.11.2 and 2.11.3):

- `\Name*{Æthelred, II}` Æthelred II; Pattern: `ÃĘthelred,II`[4]
  We have seen this name earlier.
- `\SkipIndex\Name{\AE thelred, II}` Æthelred II; Pattern: `\AEthelred,II`
  This is a new name that looks the same.

[4] With `pdflatex` / `latex`, the glyphs `ÃĘ` correspond to `\IeC{\AE}`.

- `\Name{Bo\"ethius}` Boëthius; Pattern: `Bo\"ethius`
  We introduce this new name.
- `\SkipIndex\Name{Boëthius}` Boëthius Pattern: `BoÃńthius`[5]
  This is a different name that looks the same.
- `\SkipIndex\Name{Bo{\"e}thius}` Boëthius; Pattern: `Bo{\"e}thius`
  This also is a different name that looks the same.

formatting initials

Omit spaces between initials; see Bringhurst, *Elements of Typographic Style*. If a style guide requires spaces, try thin spaces. Use `\PretagName` to sort those names (Section 2.6.3). Below we use no formatting:

```
1  \PretagName[E.\,B.]{White}%
2    {White, E.B.}
3  \begin{nameauth}
4    \< White & E.\,B. & White & >
5  \end{nameauth}
```

`\White`	E. B. White
Normal text:	E. B. White

hyphenation

English contains names from many cultures. The rules for hyphenation go to the heart of how names with non-English origins should be pronounced. With nameauth, one can use either optional hyphens or the babel / polyglossia packages to handle such names:

```
1  \newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
2   % or polyglossia: \newcommand\de[1]{\textgerman{#1}}
3  \NameAddInfo[John]{\de{Strietelmeier}}%
4    {late professor at Valparaiso University}
5  \begin{nameauth}
6    \< Striet & John & \de{Strietelmeier} & >
7  \end{nameauth}
8  \PretagName[John]{\de{Strietelmeier}}{Strietelmeier, John}
```

Not fixed:
In English, some names come from other cultures. These names, like John Strietelmeier `\SkipIndex\Name[John]{Strietelmeier}` can break badly.

Fixed with discretionary hyphens:
In English, some names come from other cultures. These names, like John Strietelmeier, `\SkipIndex\Name[John]{Strie\-tel\-meier}` could break badly.

Fixed with language packages:
In English, some names come from other cultures. These names, like John Strietelmeier, `\Striet` could break badly.

Strietelmeier (late professor at Valparaiso University) is neither pronounced nor hyphenated as "Stri-etel-meier"; rather, it is pronounced and hyphenated as "Strie-tel-meier". See Sections 2.3.4 and 2.5 when using macros in name arguments. Using babel or polyglossia likely is best.

Back to Section 1.2

[5] With `pdflatex` / `latex`, the glyphs `Ãń` correspond to `\IeC{\"e}`.

2 Detailed Usage

2.1 Package Options

One includes the nameauth package thus:

```
\usepackage[⟨option1⟩,⟨option2⟩,...,⟨optionn⟩]{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 19).**
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 version 3.0.
oldreset	Reset per-use name flags locally within the naming macros, as before version 3.3.
oldpass	When \Justindex is called, allow long / short flags to pass through, as before version 3.3.

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.

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 2.9). Otherwise \AKA uses "subsequent use" hooks.

3.0 Using the oldAKA option forces \AKA* always to print a "forename" argument in the text, as in versions before 3.0. Otherwise the current behavior of \AKA* prints in the same fashion as \FName (see Sections 2.2.2 and 2.9).

3.3 Used together, the next two options restore pre-version 3.3 handling of flags that could lead to undocumented behavior. The oldreset option causes all Boolean flags related to the prefix macros and long / short name forms to be reset locally. The new default is to reset them globally (Section 2.10.4). Likewise, the oldpass option allows the long / short flags to pass through \JustIndex instead of being reset (Section 2.6.1).

Enable / Disable Indexing

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

These options and related macros do not affect the normal use of `\index`. They 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 2.6.1.

Enable / Disable Index Sorting

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

The default allows `\PretagName` to create sort keys used with `makeindex`. The `nopretag` option disables the sorting mechanism, e.g., if a different sorting method is used with `xindy`. See Section 2.6.3.

Affect the Syntax of Names

Show / Hide Affix Commas

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

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.

In both cases, the display of commas (or the lack thereof) does not affect the appearance or the sorting of index entries.

Capitalize Entire Surnames

`normalcaps`	**Do not perform any special capitalization.**
`allcaps`	Capitalize entire surnames, e.g., 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 2.5 and 2.10.3. The simplified interface aids the embedding of control sequences in names. Section 2.3.3 deals with capitalization on a section-level and per-name basis.

Reverse Name Order

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

These three options are mutually exclusive. Section 2.3.3 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 2.3.2) are **not** the same as the `comma` option. They only affect Western names.

Typographic Post-Processing

Formatting Attributes

`noformat`	**Do not define a default format.**
`smallcaps`	First use of a main-matter name in small caps.
`italic`	First use of a main-matter name in italic.
`boldface`	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.[6] 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.[7] Post-processing does not affect the index. Sections 2.4, 2.10.1, 2.10.2, and 2.10.3 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.

Section 2.9 discusses how `\AKA` does not respect these formatting systems and uses the hooks differently. To avoid using the `formatAKA` option and `\ForceName` with `\AKA`, Section 2.6.2 shows how to use `\IndexRef` and `\Name` instead.

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

[7] This package was designed with type hierarchies and flexibility in mind. See Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60. Typographic inspiration comes from Bernhard Lohse, *Luthers Theologie* (Göttingen: Vandenhoeck & Ruprecht, 1995) and Jaroslav J. Pelikan Jr., *The Christian Tradition: A History of the Development of Doctrine*, 5 vols. (Chicago: Chicago UP, 1971–89).

Alternate or Continental Formatting

Alternate Formatting

`altformat` 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 often format surnames only, both in the text and in the index. Section 2.5 introduces the topic and should be sufficient for most users, while Section 2.10.3 goes into greater detail.

The previous methods that produced Continental formatting were rather complex compared to the current, simplified manner of doing so. Yet it is likely that these older solutions still ought to work. The error protection that prevents `\CapThis` from breaking alternately formatted names remains available to these older solutions by using `altformat` or the related macros (Section 2.5).

Feature Priority

The following table shows the relative priority of package options and macros related to indexing, capitalization, and reversing. The darker the row, the lower the priority. Those macros or options in a particular category (column) that have higher priority (row) tend to override similar macros that have lower priority in that same category.

Thus, `\IndexInactive` overrides `\JustIndex`, which overrides `\SkipIndex`; using `\SeeAlso` depends entirely on the interaction of the three others.

Indexing	Capitalization	Reversing	Name Forms, Commas, Breaks
index	normalcaps	notreversed	\ForgetThis
noindex	allcaps	allreversed	\DropAffix
\IndexActive	\AllCapsInactive	\ReverseActive	
\IndexInactive	\AllCapsActive	\ReverseInactive	
\JustIndex	\CapName	\RevName	\SubvertThis \ForceName \NoComma
\SkipIndex	\AccentCapThis	allrevcomma \RevCommaActive \RevCommaInactive	\KeepName \ForceFN \ShowComma
\SeeAlso	\CapThis	\RevComma	\KeepAffix

Back to Section 1.2

2.2 Naming Macros

In this manual we modify the formatting hooks to show first and later name uses, forcing such uses as needed (Sections 2.4 and 2.8.1). All naming macros create index entries before and after a name for when a name straddles a page break.

2.2.1 \Name and \Name*

\Name
\Name*

`\Name` displays and indexes names. It always prints the ⟨*SNN*⟩ argument. `\Name` prints the full name at the first occurrence, then usually just the ⟨*SNN*⟩ argument thereafter. `\Name*` always prints the full name:

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

3.0 In the body text, not the index, the ⟨*Alternate*⟩ argument replaces either ⟨*FNN*⟩ or, if ⟨*FNN*⟩ is absent, ⟨*Affix*⟩.[8] If both ⟨*FNN*⟩ and ⟨*Affix*⟩ are absent, then the obsolete syntax is used (Section 1.4).

```
1 \begin{nameauth}
2   \< Einstein  & Albert & Einstein        & >
3   \< Cicero    & M.T.   & Cicero          & >
4   \< Confucius &        & Confucius       & >
5   \< Miyaz     &        & Miyazaki, Hayao & >
6   \< Eliz      &        & Elizabeth, I    & >
7 \end{nameauth}
```

`\Name [Albert]{Einstein}` or `\Einstein`	Albert Einstein
`\Name*[Albert]{Einstein}` or `\LEinstein`	Albert Einstein
`\Name [Albert]{Einstein}` or `\Einstein`	Einstein
`\Name [M.T.]{Cicero}` or `\Cicero`	M.T. Cicero
`\Name*[M.T.]{Cicero}[Marcus Tullius]`	Marcus Tullius Cicero
`\Name [M.T.]{Cicero}` or `\Cicero`	Cicero
`\Name {Confucius}`, `\Confucius`	Confucius
Same for all variants; no ⟨*Affix*⟩ or ⟨*Alternate*⟩.	Confucius
`\Name {Miyazaki, Hayao}` or `\Miyaz`	Miyazaki Hayao
`\Name*{Miyazaki, Hayao}[Sensei]`	Miyazaki Sensei
`\Name {Miyazaki, Hayao}` or `\Miyaz`	Miyazaki
`\Name {Elizabeth, I}` or `\Eliz`	Elizabeth I
`\Name*{Elizabeth, I}` or `\LEliz`	Elizabeth I
`\Name {Elizabeth, I}` or `\Eliz`	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 alternate forename is not shown in subsequent short name references e.g., `\Cicero[Marcus Tullius]` Cicero.

[8] If ⟨*Alternate*⟩ is `\ignorespaces`, the Western long form of `\Name` looks like the short form. "Native" Eastern and ancient forms would have an extra trailing space.

2.2.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print personal names only in subsequent name uses. They print full names for first uses. These synonyms let one add an F either to \Name or \Name* to get the same effect:

```
\FName [⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
\FName*[⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate⟩]
```

\ForceFN
3.0

These macros work with both Eastern and Western names, but to get an Eastern personal name, one must precede these macros with \ForceFN. See also Sections 2.3.4 and 2.8.1 on how to vary some of the forms below:

\FName[Albert]{Einstein} or \SEinstein	Albert
\FName[M.T.]{Cicero}[Marcus Tullius] or \SCicero[Marcus Tullius]	Marcus Tullius
\FName{Confucius} or \SConfucius	Confucius
\FName{Miyazaki, Hayao} or \SMiyaz	Miyazaki
\ForceFN\FName{Miyazaki, Hayao} or \ForceFN\SMiyaz	Hayao
\ForceFN\FName{Miyazaki, Hayao}[Sensei] or \ForceFN\SMiyaz[Sensei]	Sensei
\FName{Elizabeth, I} or \SEliz	Elizabeth
\ForceFN\SEliz[Good Queen Bess]	Good Queen Bess

The ⟨*Alternate*⟩ argument replaces forenames in the text, which strongly shapes the use of \FName.[9] We recap what we saw on page 9, emphasizing forenames:

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

These share name patterns: \SCSL C.S., \SLewis Clive Staples; \SChes Chesley B. Sullenberger III, \SSully Sully; \SMiyaz Miyazaki, \SMSens Miyazaki.

Equivalents: \SCSL C.S., \SLewis[C.S.] C.S.; \SSully Sully, \SChes[Sully] Sully; \ForceFN\SMSens Sensei, \ForceFN\SMiyaz[Sensei]: Sensei.

These fail: \SCSL[Jack]: C.S.[Jack]; \SSully[Chesley]: Sully[Chesley]; and \ForceFN\SMSens[Hayao]: Sensei[Hayao]. Whenever ⟨*arg4*⟩ of the **nameauth** environment is used, the respective shorthands cannot take optional arguments.

Back to Section 1.2

[9] If ⟨*Alternate*⟩ is \ignorespaces, the Western long form of \FName looks like the short form of \Name, while the Western short form of \FName acts like \leavevmode and prints nothing. "Native" Eastern and ancient forms would have an extra trailing space.

2.2.3 Variant Names

3.1 This section explains how to manage more complicated variants, which gives one the skills needed to implement a name authority. We draw from Sections 2.4, 2.6.2, 2.6.3, 2.8.1, and 2.9. One might want to consult those sections also.

variant forenames

We begin with the easier kind of variant names, namely, variant forenames indexed under a canonical name entry:

```
\begin{nameauth}
  \< Tyson & Mike & Tyson & >
  \< Iron & Mike & Tyson & Iron Mike >
\end{nameauth}
```

Same pattern: `\Iron` Iron Mike Tyson `\LTyson` Mike Tyson
`\SIron` Iron Mike `\STyson` Mike

Since Iron Mike Tyson is indexed as "Tyson, Mike" throughout the document, we can use `\IndexRef{Iron Mike}{Tyson, Mike}` with no output in the text or `\AKA[Mike]{Tyson}{Iron Mike}` Iron Mike to print a name. Both create the cross-reference "Iron Mike *see* Tyson, Mike" in the index.

variant surnames

Variant family names are more complicated than variant personal names. For surname variants, one can use the following method to get fairly good results, depending on the trade-offs that one wishes to accept:

```
\begin{nameauth}
  \< DuBois    & W.E.B. & Du~Bois & >
  \< AltDuBois & W.E.B. & DuBois  & >
\end{nameauth}
\PretagName[W.E.B.]{Du~Bois}{Dubois, W.E.B.}
```

1. We decide the canonical name form: `\DuBois` W.E.B. Du Bois.
2. Both `\Name[W.E.B.]{Du Bois}` and `\Name[W.E.B.]{DuBois}` have the pattern "`W.E.B.!DuBois`" (Section 1.5). Here we use `Du~Bois` as the argument because we want no breaks, giving us "`W.E.B.!Du~Bois`".
3. We set the sort key for both names to be `{Dubois, W.E.B.}`. If it were of the form `{Du Bois, W.E.B.}`, they would sort differently (Section 2.6.3). One must check a style manual for proper sorting.
4. Instead of using `\SkipIndex\AltDuBois` many times, we create a cross-reference in the preamble so that no page entry for the alternate form will occur in the index:

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

5. We can use `\JustIndex\DuBois\AltDuBois` W.E.B. DuBois, keep full stop detection, and check if the name straddles a page break in order to append `\JustIndex\DuBois` if needed.
6. If we create a macro like the one below, we lose full stop detection but then we do not have to check if the name straddles a page break. Normally, the name macros create two index entries each in order to handle this issue automatically:

   ```
   \newcommand\NewDuBois%
     {\JustIndex\DuBois\AltDubois\JustIndex\DuBois}
   ```

Example Name Authority

Below are a couple of names from a name authority created for a translation of *De Diaconis et Diaconissis Veteris Ecclesiae Liber Commentarius* by Caspar Ziegler, of which the present author was the editor.[10]

Constructing that name authority was a challenge. In order to get the names right — the deceased translator unfortunately had left them in abbreviated Latin, as well as leaving many place names in Latin or translating them incorrectly — the present author used the following sources, among several others:

- CERL Thesaurus: https://data.cerl.org/thesaurus/_search
- Virtual International Authority File: http://viaf.org/
- EDIT16: http://edit16.iccu.sbn.it/web_iccu/ehome.htm
- WorldCat: https://www.worldcat.org/
- An older version of Graesse, *Orbis Latinus*: http://www.columbia.edu/acis/ets/Graesse/contents.html

This author followed the scholarly standards for determining the canonical name forms and used the alternate names (which were the ones actually in the original text) to refer to the canonical forms. I just translated all the place-names.

Below we have candidates for sorting with `\PretagName` (Section 2.6.3) and potential use of `\CapThis` (Section 2.3.4). After using `\IndexRef` with a particular name, using `\Name` with that same name will not create a page reference from that point onward (Section 2.6.2).

```
\PretagName[Jacques]{De~Pamele}{Depamele, Jacques}
\Name[Jacques]{De~Pamele}[Jacques de~Joigny]
\IndexRef[Jacobus]{Pamelius}{De~Pamele, Jacques}
\Name[Jacobus]{Pamelius}

\PretagName[Giovanni]{d'Andrea}{Dandrea, Giovanni}
\Name[Giovanni]{d'Andrea}
\IndexRef[Ioannes]{Andreae}{d'Andrea, Giovanni}
\Name[Ioannes]{Andreae}
```

Canonical Name	Alternate Name
Jacques de Joigny De Pamele	Jacobus Pamelius
Giovanni d'Andrea	Ioannes Andreae

D'Andrea `\CapThis\Name[Giovanni]{d'Andrea}` can be used at the beginning of a sentence. `\Name[Jacques]{De~Pamele}` gives De Pamele.

Back to Section 1.2

[10] The book, *The Diaconate of the Ancient and Medieval Church*, originally was typeset using LaTeX, but had to be converted to a different format. Using LaTeX, the present author has published Charles P. Schaum and Albert B. Collver III, *Breath of God, Yet Work of Man: Scripture, Philosophy, Dialogue, and Conflict* (St. Louis: Concordia Publishing House, 2019).

2.3 Language Topics

This section looks at how nameauth addresses grammar, usage, and cultural standards. The concept of comma-delimited affixes dominates much of this section.

2.3.1 Affixes Require Commas

A comma is required to separate a Western surname and affix, an Eastern family name and personal name, and an ancient name and affix. Yet we must take care because an example like `\Name{\textsc{a Name, Problem}}` will halt LaTeX with errors (Section 2.5). Spaces around the comma are ignored (Section 1.6).

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

Western names with affixes must use the comma-delimited syntax. Using the obsolete syntax, `\SkipIndex\Name[Oskar]{Hammerstein}[II]` produces II Hammerstein which is an error. See also (Section 2.9).

`\KeepAffix` In the text only, `\KeepAffix` turns the space **between** ⟨*SNN*⟩ and ⟨*Affix*⟩ 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` **3.1** In the text only, `\KeepName` turns all spaces **between** name elements ⟨*FNN*⟩, ⟨*SNN*⟩, and ⟨*Affix*⟩ into non-breaking spaces; `\KeepName\LJWG[von]` von Goethe will not break. This macro does not alter spaces **within** name elements like ⟨*FNN*⟩ (French or German forenames) and ⟨*SNN*⟩ (Spanish surnames). Both `\KeepAffix` and `\KeepName` can affect nameauth macros that print in the text.

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

`\ShowComma` `\NoComma` **2.6** `\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. Neither of these macros affect the use of `\RevComma`, which always prints a comma.

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

Back to Section 1.2

2.3.2 Listing Western names by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma
3.0

In addition to the options for reversed comma listing (Section 2.1), the macros `\ReverseCommaActive` and `\ReverseCommaInactive` function the same way with blocks of text. They all override `\RevComma`. These all reorder long Western name forms (via `\Name*` and the like). The first two are broad toggles, while the third works on a per-name basis. These macros only affect Western and "non-native" Eastern name forms.

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

\global

Both `\ReverseCommaActive` and `\ReverseCommaInactive` can be used either as a pair or singly within a local scope. Use `\global` to force a global effect.

Back to Section 1.2

2.3.3 Eastern Names

"non-native"

One produces a "non-native" Eastern name in the text by reversing a Western without ⟨*Affix*⟩ using `\RevName`:

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

The index entry of this name form looks like ⟨*SNN*⟩, ⟨*FNN*⟩ (including the comma). This is a Western index entry. This form is used also for Hungarian names, e.g.: `\RevName\Name[Frenec]{Molnár}` Molnár Frenec†, Molnár†.

"native"

In contrast, "native" Eastern names use either comma-delimited syntax or the obsolete syntax. They have Eastern-form index entries ⟨*SNN*⟩ ⟨*Affix/Alternate*⟩ (no comma). The new syntax permits alternate names; the obsolete does not. These forms work also with ancient and medieval names:

```
\Name{⟨SNN, Affix⟩}[⟨Alternate⟩]   % new syntax
\Name{⟨SNN⟩}[⟨Alternate⟩]          % obsolete syntax
```

avoid error

People can make mistakes that these forms help one to avoid. For example, in an otherwise excellent German-language history textbook series, one finds an index entry for "Yat-sen, Sun". It should be "Sun Yat-sen".[11] The form `\Name{Sun, Yat-sen}` Sun ensures the correct entry.

\ReverseActive
\ReverseInactive
\RevName

In addition to the options for reversing (Section 2.1), `\ReverseActive` and `\ReverseInactive` reverse name order for blocks of text. These all override the use of `\RevName`, which reverses once per name. These macros do not affect the

[11] See Immanuel Geiss, *Personen: Die biographische Dimension der Weltgeschichte*, Geschichte Griffbereit vol. 2 (Munich: Wissen Media Verlag, 2002), 720. Errors arising from cultural differences and basic mistakes give justification for the design of nameauth.

index. They work also with \AKA and its derivatives. The reverse mechanism shows only in full names, but it does not force full names. “Non-native” forms are shown by a dagger (†) in the next table:

	unchanged	\RevName
\LNoguchi	Hideyo Noguchi	Noguchi Hideyo†
\LNoguchi[Doctor]	Doctor Noguchi	———
\LNoguchi[Sensei]	———	Noguchi Sensei†
\Noguchi	Noguchi	Noguchi†
\SNoguchi	Hideyo	Hideyo†
\LYamt	Yamamoto Isoroku	Isoroku Yamamoto
\LYamt[Admiral]	———	Admiral Yamamoto
\Yamt	Yamamoto	Yamamoto
\SYamt	Yamamoto	Yamamoto
\ForceFN\SYamt	Isoroku	Isoroku

\global Both \ReverseActive and \ReverseInactive can be used either as a pair or singly within an explicitly local scope. Use \global to force a global effect.

\AllCapsActive \AllCapsInactive \CapName In addition to the options for capitalizing (Section 2.1), \AllCapsActive and \AllCapsInactive work for blocks of text. All override \CapName, which works once per name. These capitalize ⟨*SNN*⟩ in the body text only. They also work with \AKA and friends. For caps in the text and index see Sections 2.5 and 2.10.3. We show “non-native” Eastern forms with a dagger (†) and the old syntax with a double dagger(‡).

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

\global Both \AllCapsActive and \AllCapsInactive can be used either as a pair or singly within an explicitly local scope. Use \global to force a global effect.

Back to Section 1.2

2.3.4 Particles, Medieval Names, and Ancient Names

cap rules English names with particles *de*, *de la*, *d'*, *von*, *van*, and *ten* often keep them with the last name, using varied capitalization.[12] *Le*, *La*, and *L'* always are capitalized unless preceded by *de*. See also Sections 1.3.2, 1.5, 2.2.3, and 2.5.

non-breaking spaces We recommend inserting a tilde (active character for a non-breaking space) or \nobreakspace between some particles and names to prevent bad breaks, sorting them with \PretagName (Section 2.6.3). Some particles look similar: *L'* (L+apostrophe) and *d'* (d+apostrophe) are two separate glyphs each. In contrast, *Ľ* (L+caron) and *ď* (d+caron) are one Unicode glyph each (Section 2.11.2).

[12] According to [Mulvany, 165f.] and the *Chicago Manual of Style*.

\CapThis

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

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

3.2

\CapName overrides the ⟨*SNN*⟩ created by \CapThis. \CapThis should work with all of the Unicode characters available in the T1 encoding (its mechanism is explained in Section 2.11.2 and on page 78). For a broader set of Unicode characters, consider using xelatex and lualatex.

surname variants

For another example, we mention poet e.e. cummings. One can have formatted name caps and inflections, e.g.: "Cummings's motif of the goat-footed balloon man has underlying sexual themes that nevertheless have a childish facade." The easiest way to do that is from Section 2.6.2:

```
\ExcludeName[e.e.]{cummings's}
...
\SubvertThis\CapThis\Name[e.e.]{cummings's}%
\IndexName[e.e.]{cummings} ..... Cummings's
```

One must use \SubvertThis only for the first use to avoid "E.e. Cummings's"; all name elements are capped with \CapThis. Using \ExcludeName keeps one from having to use \SkipIndex every time. With nameauth we can use both simple and complex solutions to name variation. See also Section 2.2.3.

Section 2.5 explains how to use \CapThis with alternate formatting when using macros in name arguments. Page 38 describes how automation lends itself to Continental (French, German, etc.) formats and grammatical inflections.

\AccentCapThis

3.0

If one uses this package on a system that does not handle Unicode, one can use \AccentCapThis instead of \CapThis to handle active initial characters. Otherwise, one should not need to use \AccentCapThis.

Examples

medieval names

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

```
1 \PretagName{Thomas, à~Kempis}{Thomas Akempis}  % medieval
2 \PretagName[Thomas]{à~Kempis}{Akempis, Thomas} % Western
3 \IndexRef[Thomas]{à~Kempis}{Thomas à~Kempis}    % xref
4 \ExcludeName{Thomas,\‘a~Kempis} % alternate form excluded
5 \begin{nameauth}
6   \< KempMed & & Thomas,   à~Kempis & >         % medieval
7   \< KempW     & Thomas  & à~Kempis & >         % Western
8 \end{nameauth}
```

3.1

The medieval forms Thomas à Kempis and Thomas are indexed as "Thomas à Kempis." The place name \ForceFN\SKempMed "à Kempis" (Latin for *von Kempen*) technically is not a Western surname. À Kempis \CapThis\ForceFN\SKempMed starts a sentence. Thomas à Kempis \Name{Thomas,\‘a~Kempis} is different. À Kempis is \CapThis\SubvertThis\ForceFN\FName{Thomas,\‘a~Kempis}. One should use \PretagName to sort the index entry (Section 2.6.3). We excluded this alternate form (Section 2.6.2).

Western forms like `\KempW`: Thomas à Kempis are very different from medieval forms and create different index entries. `\CapThis\KempW` gives "À Kempis" in the text and "à Kempis, Thomas" in the index.

Above, we created a cross-reference from the Western form to the medieval form, preventing page entries (Section 2.6.2). If we sorted the cross-reference using `\PretagName[Thomas]{à~Kempis}{a Kempis, Thomas}`, it would precede `aardvark`. We use `\PretagName[Thomas]{à~Kempis}{Akempis, Thomas}`, which sorts the cross-reference between `ajar` and `alkaline`. One should check a style manual for correct sorting (Section 2.6.3).

ancient names

Ancient contexts may or may not bind particles to surnames. The ⟨*alternate*⟩ argument, `\PretagName`, and `\TagName` address this (Sections 2.6.3, 2.6.4).

The next examples do not use the formatting conventions of this manual and sometimes hide details that are specific to this manual in order to keep things simple and reflect normal document usage. See the `dtx` source code for more information. First we use variants with ⟨*alternate*⟩:[13]

```
\NameAddInfo{Demetrius, I}{ Soter}
\PretagName{Demetrius, I}{Demetrius 1}
\TagName{Demetrius, I}{ Soter, king}
\begin{nameauth}
  \< Dem & & Demetrius, I & >
\end{nameauth}
```

`\Dem[I Soter]`	Demetrius I Soter
`\LDem`	Demetrius I
`\Dem`	Demetrius

For a more automated approach, we can use "text tags" in the formatting macros (see Sections 2.7, 2.10.2).

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

`\ForgetThis\Dem`	Demetrius I Soter
`\LDem`	Demetrius I
`\Dem`	Demetrius

Roman names

The Roman naming system does present some challenges. As long as we do not use `\CapThis`, we do not need alternate formatting (Section 2.5). Earlier we treated Marcus Tullius Cicero as a Western name. Now we handle Roman names more properly.

[13] Copies of these examples are in `examples.tex`, collocated with this manual.

We have a *praenomen*, a personal name. Then we have a *nomen*, a clan name. Finally, we would have a *cognomen*, a nickname, except it became hereditary to denote clan branches. Added to that are *agnomina*, affixed names.

popular works

Popular sources tend to treat the *cognomen* as we might a surname, with the indexed form: ⟨*cognomen*⟩ ⟨*agnomen*⟩, ⟨*praenomen*⟩ ⟨*nomen*⟩.[14] We want all names in the index, so we define macros in ⟨*FNN*⟩ and ⟨*SNN*⟩ that expand one or two components: *praenomen* and *nomen*, *cognomen* and *agnomen*. We begin by defining a name with macros using `\noexpand` to prevent error:

```
\begin{nameauth}
  \< Scipio & \noexpand\SCIPi & \noexpand\SCIPii & >
\end{nameauth}
\PretagName[\noexpand\SCIPi]{\noexpand\SCIPii}{Scipio Africanus}
```

We define the flags and macros by which the name will change. The global state of `\NoGens` and`\NoAgnomen` determine the index form. The local scope in the formatting hooks allows changes that are reset when exiting that scope. The logic is inverted; false prints long, true prints short:

```
\newif\ifSkipGens
\newif\ifNoGens
\newif\ifSkipAgnomen
\newif\ifNoAgnomen
\newcommand*\SCIPi{\ifNoGens Publius\else
                    Publius Cornelius\fi}
\newcommand*\SCIPii{\ifNoAgnomen Scipio\else
                     Scipio Africanus\fi}
\newcommand*\ScipioOnly{\SkipAgnomentrue\Scipio}
\renewcommand*\NamesFormat[1]%^^A
  {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
  \global\SkipGensfalse\global\SkipAgnomenfalse}
\renewcommand*\MainNameHook[1]%^^A
  {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
  \global\SkipGensfalse\global\SkipAgnomenfalse}
```

> Publius Cornelius Scipio `\ScipioOnly` was born around 236 BC into the Scipio branch of the Cornelius clan, one of six large patrician clans. Scipio `\ScipioOnly` rose to military fame during the Second Punic War. Thereafter he was known as Scipio Africanus `\Scipio`.

An advantage of the popular format is that one can drop both *praenomen* and *nomen* automatically in subsequent uses. Yet in any case, one can define helper macros to change Boolean flags. The raw index entry is fairly lengthy by necessity, governed by the global state of the Boolean flags, and expanding to:

> Scipio Africanus=Scipio Africanus, Publius Cornelius|hyperpage

scholarly works

The *Oxford Classical Dictionary* and other scholarly sources index under the *nomen*. That requires a similar approach, but it moves the *nomen* from ⟨*FNN*⟩ to ⟨*SNN*⟩. Although we will not index the name, we will show how to set up Scipio Africanus to work in that alternate configuration.

[14] See Geiss, *Geschichte Griffbereit*; Kinder and Hilgemann, *dtv-Atlas zur Weltgeschichte*, 2 vols., 29th printing (1964; Munich: Deutscher Taschenbuch Verlag, 1993). For further resources see: http://books.infotoday.com/books/Indexing-names.shtml. See also https://en.wikipedia.org/wiki/Roman_naming_conventions.

We keep the Boolean flags as previously created. We redefine the name in the following manner:

```
\begin{nameauth}
  \< OScipio & Publius & \noexpand\CSA & >
\end{nameauth}
\PretagName[Publius]{\noexpand\CSA}{Cornelius Scipio Africanus}
```

We use a nested conditional in ⟨*SNN*⟩. The default still is to show all names so that they can be indexed that way. This time we decided to index under the popular form instead of the scholarly one, so we exclude the scholarly form:

```
\newcommand*\CSA{\ifNoGens
                   \ifNoAgnomen
                   Scipio\else
                   Scipio Africanus\fi
                 \else\ifNoAgnomen
                   Cornelius Scipio\else
                   Cornelius Scipio Africanus\fi\fi}
\ExcludeName[Publius]{\noexpand\CSA}
```

The scholarly form has a different name pattern, so it is not compatible with the popular version. Nevertheless, we show what the raw index entry of the scholarly form would be. We include some of the more meaningful forms of both versions:

`\ShowPattern[Publius]{\noexpand\CSA}`:
Publius!\noexpand\CSA

`\ShowIdxPageref[Publius]{\noexpand\CSA}`:
Cornelius Scipio Africanus=Cornelius Scipio Africanus, Publius|hyperpage

First use:

`\OScipio`: Publius Cornelius Scipio Africanus
`\Scipio` : Publius Cornelius Scipio Africanus

Subsequent use:

`\OScipio`: Cornelius Scipio Africanus
`\SkipGenstrue\OScipio`: Scipio Africanus
`\Scipio` : Scipio Africanus

Subsequent use, full, no *agnomen*:

`\SkipAgnomentrue\LOScipio`: Publius Cornelius Scipio
`\SkipAgnomentrue\LScipio` : Publius Cornelius Scipio

Subsequent use, shortest forms:

`\SkipAgnomentrue\OScipio`: Cornelius Scipio
`\SkipGenstrue\SkipAgnomentrue\OScipio`: Scipio
`\SkipAgnomentrue\Scipio` : Scipio

Subsequent use, personal name:

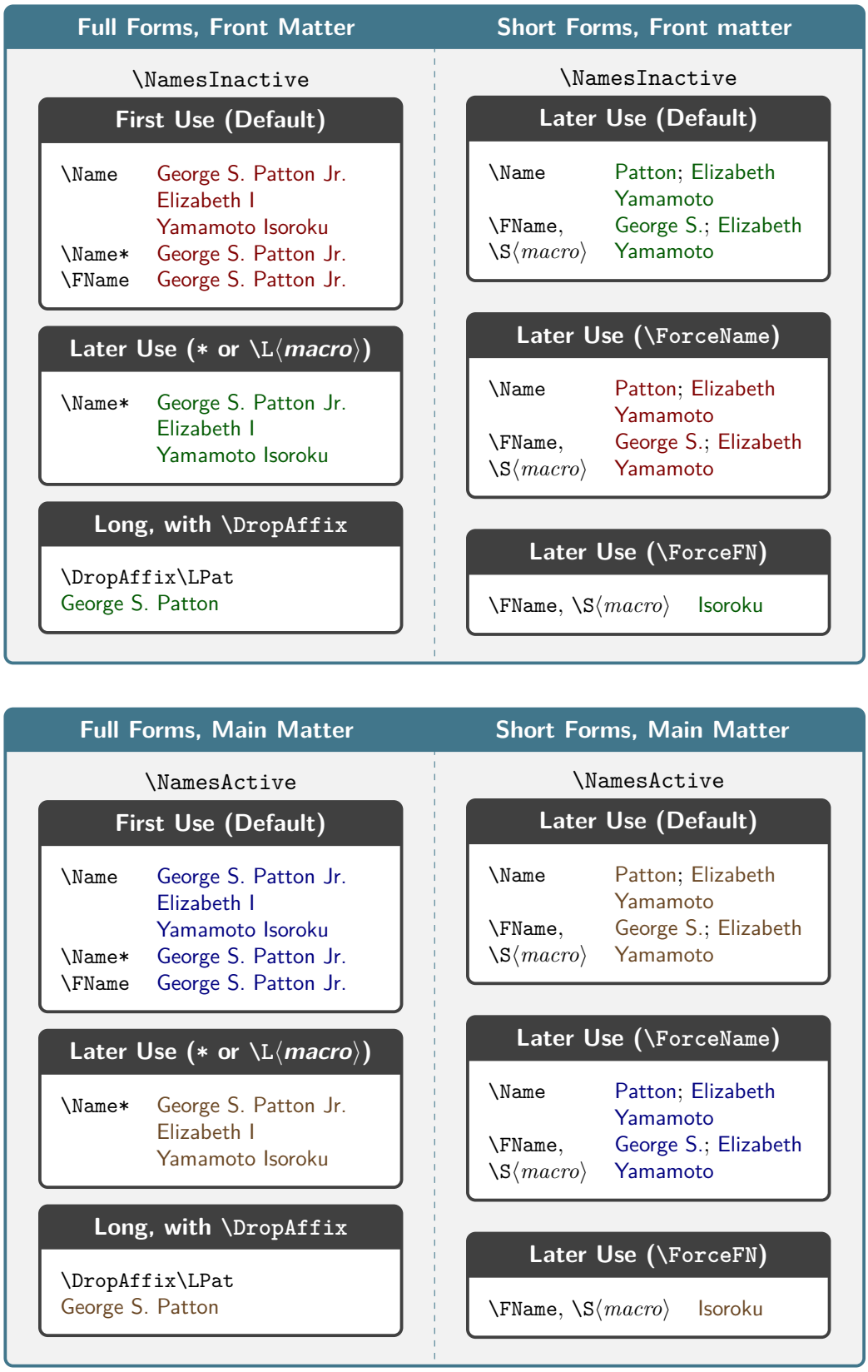
`\SOScipio`: Publius
`\SkipGenstrue\SScipio` : Publius

See Sections 1.6, 2.5, and 2.10 for more guidance on avoiding errors when using name arguments that contain macros.

Back to Section 1.2

2.4 Basic Formatting

Below are many of the forms and formats that names can have:



These formatting features of nameauth can work with name control macros (Section 2.8.1) in, for example, beamer overlays to define consistently the context and outcome of how names appear. There are two kinds of formatting at work that interact with each other:

1. **Syntactic Formatting:** Reversing and caps normally occur only in the body text, not the index. Yet macros in name arguments affect both text and index.
2. **Name Post-Processing:** Hook macros apply formatting only to the printed form of a name after parsing. See also Section 2.10.3.

\NamesFormat
\MainNameHook
\FrontNamesFormat
\FrontNameHook

2.5

Independent "main-matter" and "front-matter" systems are used to 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 uses and \FrontNameHook for subsequent uses. The alwaysformat option causes only \NamesFormat and \FrontNamesFormat to be used (cf. Section 1.5).[15]

\NamesActive
\NamesInactive

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

\global

These two macros can be used explicitly as a pair or singly within an explicit local scope. Use \global to force a global effect.

The two formatting systems are distinct, useful for front matter and main matter, text and footnotes, etc. We show this with different colors:

```
\colorlet{nared}{red!50!black}
\colorlet{nagreen}{green!35!black}
\colorlet{nablue}{blue!50!black}
\colorlet{nabrown}{brown!55!black}
\renewcommand*\FrontNamesFormat[1]{\color{nared}\sffamily #1}
\renewcommand*\FrontNameHook[1]{\color{nagreen}\sffamily #1}
\renewcommand*\NamesFormat[1]{\color{nablue}\sffamily #1}
\renewcommand*\MainNameHook[1]{\color{nabrown}\sffamily #1}
```

Front-matter system:	\NamesInactive
\Name[Rudolph]{Carnap}	Rudolph Carnap
\Name[Rudolph]{Carnap}	Carnap
\Name[Nicolas]{Malebranche}	Nicolas Malebranche
\Name[Nicolas]{Malebranche}	Malebranche

Main-matter system:	\NamesActive
\Name[Rudolph]{Carnap}	Rudolph Carnap
\Name[Rudolph]{Carnap}	Carnap
\Name[Nicolas]{Malebranche}	Nicolas Malebranche
\Name[Nicolas]{Malebranche}	Malebranche

[15] The names of these macros may seem poorly conceived. When starting work on this package, this author was ignorant of the breadth of how names might be handled. Designed to meet the needs of a master's thesis, this package has evolved to meet the needs of several published works. At one time, \NamesFormat was the only macro that did any formatting. The rest came later. A certain degree of cargo cult programming arose, to be corrected in the 3.0 series of nameauth.

\ForceName

3.1 Use this prefix macro to force "first use" formatting for the next `\Name`, etc. This will not force a full name reference like `\ForgetThis`. One must use the `formatAKA` option when using this with `\AKA`, etc. We show `\ForceName` in Sections 2.8.1, 2.9, and 2.10.2.

alwaysformat

Below we simulate the `alwaysformat` option by manipulating the package internals. Using first-use hooks will not force full name references. This option made more sense when `\NamesFormat` was the only formatting hook.

- 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.

hook caveats

The internal hook dispatcher calls the formatting hooks using the pattern `\bgroup`⟨*Hook*⟩`{#1}\egroup`. Thus one can use, e.g., `\itshape` in a local scope. One also can use macros that take one argument (cf. Section 2.10.3), e.g., `\renewcommand*\NamesFormat{\sffamily\color{nablue}\textit}` will create the forms *Albert Einstein* and Einstein.

applied to footnotes

The independent systems or "species" of names fit independent text elements, like front matter or even footnotes. Names in the body text, such as John Maynard Keynes, also affect names in the footnotes.[16] In footnote 16 `\MainNameHook` is called instead of `\NamesFormat` because Keynes already had occurred above.

If we wanted to format names differently in the footnotes than in the body text, an easy way to do that is to use the front-matter system. For example:

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

When we create another footnote, we see very different results.[17] Footnote 17 shows a completely independent formatting. One also can synchronize the two systems with `\ForgetThis` and `\SubvertThis` (Section 2.8 and its subsections).

To finish this example, we change footnotes back to normal:

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

Of cource, one can force long and short forms as needed (Section 2.8.1). Yet the main point of nameauth is to do the complex work once, then use that in automated fashion for the rest of the document.

Back to Section 1.2

[16] We have Keynes from `\Name[John Maynard]{Keynes}` instead of John Maynard Keynes.
[17] We have John Maynard Keynes from `\Name[John Maynard]{Keynes}`, then Keynes.

2.5 Alternate Formatting

The formatting hooks only affect names in the body text. Continental formatting occurs in both the text and in the index. One needs to format those names with macros in the name arguments. The basic way formats names in both text and index. The advanced way allows changes in the text, but keeps the index consistent.

2.5.1 Basic Features

Section 1.6 showed us that changing a control sequence will change the index entry of a name, even if one cannot see differences on the page. Alternate formatting helps one avert spurious index entries.

Using, e.g., `\Name{\textsc{a Name, Problem}}` will halt LaTeX because the comma tries to break `\textsc` and its argument into two elements. We fix that with: `\Name{\textsc{a Name}, \textsc{Problem}}`. Yet `\CapThis` still needs alternate formatting, given that `\textsc` is robust (Section 2.5.2).[18]

`\AltFormatActive` Both the `altformat` option and `\AltFormatActive` enable and activate alternate formatting. Both cause `\CapThis` to work via `\AltCaps` instead of the normal way. `\AltFormatActive` countermands `\AltFormatActive*`.

- **Enabled** means that the alternate formatting mechanism inhibits the normal behavior of `\CapThis`.
- **Activated** means that `\textSC` and other alternate formatting macros (see below) format their arguments. When deactivated, they do not format their arguments.

3.1 At the start of this section we used `\AltFormatActive` to enable alternate formatting and "switch on" the alternate formatting macros. That is the basic set of conditions for the simple use of alternate formatting.

`\AltFormatActive*` The starred form `\AltFormatActive*` enables alternate formatting but deactivates the special formatting macros, preventing them from changing their arguments. It countermands both the `altformat` option and `\AltFormatActive`. It causes `\CapThis` only to work via `\AltCaps`.

`\AltFormatInactive` To both disable alternate formatting and deactivate the alternate formatting macros, use `\AltFormatInactive`. This reverts globally to standard formatting and the normal function of `\CapThis`.

	Enabled	Activated
`\AltFormatActive`	■	■
`\AltFormatActive*`	■	
`\AltFormatInactive`		

On the next page we describe the formatting macros that are built in to nameauth in order to use the basic features of alternate formatting and provide a foundation for the advanced features. One should use `\PretagName` (Section 2.6.3) to sort the related index entries for these names.

[18] Pre-version 3.1 methods of Continental formatting should work if one uses the `altformat` option or `\AltFormatActive` to protect against the default behavior of `\CapThis`.

\textSC
\textIT
\textBF
\textUC

Continental formatting can be as simple as using the short macro \textSC. Three other macros also implement alternate formatting. These macros make changes only when alternate formatting is active. We sort the index entry with \PretagName and demonstrate the formatting.

```
\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

Since we switch to Latin Modern Sans in the formatting hooks, the switch to small caps in Garbo forces a substitution to Latin Modern Roman. This action varies with the font being used.

Using basic alternate formatting, these macros **always format their arguments** with the altformat option or \AltFormatActive. Likewise, they **never format their arguments** when \AltFormatActive* is used. To change the formatting of the name arguments, one must use the advanced features. Whenever a naming macro writes to the index, the formatting macros must be in the same Boolean state to avoid spurious index entries. The next section explains more.

As with normal formatting, \CapName interacts with alternate formatting only in the text. Thus Greta GARBO instead of Garbo. \RevComma likewise gives *Lovelace*, Ada. \RevName produces Ieyasu TOKUGAWA.

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.

```
\PretagName[John David]{\textSC{Rockefeller},\textSC{III}}
  {Rockefeller, John David 3}
\PretagName{\textUC{Fukuyama}, Takeshi}{Fukuyama Takeshi}
\begin{nameauth}
  \< JRIII & John David & \textSC{Rockefeller},\textSC{III} & >
  \< Fukuyama  & & \textUC{Fukuyama}, Takeshi &         >
  \< OFukuyama & & \textUC{Fukuyama}          & Takeshi >
\end{nameauth}
```

From above we get John David Rockefeller III, then Rockefeller. 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.

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

Only the new syntax allows one to use alternate names in the text (Section 2.2.2). For example, "\LFukuyama[Sensei] FUKUYAMA Sensei wrote *Nihon Fukuin Rūteru Kyōkai Shi* in 1954, after studying in the US in the 1930s."

2.5.2 Advanced Features

A more complex version of alternate formatting allows us to make formatting and other changes in the text while keeping the index consistent. In order to do this, we will be using `\textSC`, `\textIT`, `\textBF`, and `\textUC` with `\noexpand` and special triggering macros. Below we briefly see the difference:

```
\Name[Martin]{\textSC{Luther}}              % basic alternate formatting
\Name[Martin]{\noexpand\textSC{Luther}}     % advanced version
```

The reason for this approach is that indexing operations occur outside the formatting hooks, never within the hooks, and `\noexpand` keeps the two separate.

> Using `\noexpand` is key to consistent index entries.

`\AltCaps` `\CapThis` causes `\AltCaps` to cap its argument only in a formatting hook. It is enabled whenever alternate formatting is enabled. `\AltCaps` works independently of `\AltOn` and `\AltOff`:

`\noexpand\AltCaps{`⟨*Arg*⟩`}`

In the example below we redefine `\MainNameHook` to suppress formatting:

```
\renewcommand*\MainNameHook%
  {\color{nabrown}\sffamily\AltOff}% we match the manual

\IndexInactive
What's in a \Name{\noexpand\AltCaps{a} Name}?
\CapThis\Name{\noexpand\AltCaps{a} Name} smells not,
but a rose does. We avoid \Name{
  \noexpand\textSC{\noexpand\AltCaps{a} Name},
  \noexpand\textSC{Problem}}.
\CapThis\Name*{
  \noexpand\textSC{\noexpand\AltCaps{a} Name},
  \noexpand\textSC{Problem}} will not occur,
 even if it smells like a rose.
```

What's in a a Name? A Name smells not, but a rose does. We avoid A Name Problem. A Name Problem will not occur, even if it smells like a rose.

`\AltOff` Like a manual automobile clutch and gearbox, `\AltOff` deactivates `\textSC`, `\textBF`, `\textIT`, and `\textUC` only in a formatting hook.

`\AltOn` `\AltOn` activates `\textSC`, `\textBF`, `\textIT`, and `\textUC` only in a formatting hook. To summarize:

- `\AltFormatActive` and `\AltFormatActive*` set global states.
- `\AltFormatActive` causes formatting in the text and index, as well as forcing the use of `\AltCaps`.
- With `\AltFormatActive*` inhibits formatting, but still requires one to use `\AltCaps`.
- `\AltOn` and `\AltOff` change local state only in the formatting hooks.
- The user adds `\AltOn` and `\AltOff` to the hooks as needed.
- The actual formatting happens via macros in the name arguments.

Keeping the \MainNameHook example above, we have:

```
\begin{nameauth}
  \< Luth & Martin & \noexpand\textSC{Luther} & >
\end{nameauth}
\PretagName[Martin]{\noexpand\textSC{Luther}}{Luther, Martin}
```

We first mention \Luth Martin Luther. Then again, \Luth Luther. Medieval Italian differs from modern Italian with respect to particles. Below the index entry should be "Medici, Catherine de' " instead of "de Medici, Catherine":

```
\begin{nameauth}
  \< Cath & Catherine \noexpand\AltCaps{d}e'
          & \noexpand\textSC{Medici} & >
\end{nameauth}
\PretagName[Catherine \noexpand\AltCaps{d}e']
            {\noexpand\textSC{Medici}}{Medici, Catherine de}
```

This gives us Catherine de' Medici and Medici. To get De' Medici and De' Medici in the text, use \CapThis\LCath[\noexpand\AltCaps{d}e'].

name inflections

We can use alternate formatting for grammatical inflections (cf. Section 2.10.3). We tell the same set of lies that we did on page 29. \DoGentrue occurs only in the formatting hook, thereby keeping the index entries consistent:[19]

```
\newif\ifGenitive
\newif\ifDoGen
\renewcommand*\NamesFormat[1]
  {\ifGenitive\DoGentrue\fi#1\global\Genitivefalse}
\renewcommand*\MainNameHook[1]
  {\ifGenitive\DoGentrue\fi\AltOff#1\global\Genitivefalse}
\begin{nameauth}
  \< Jeff & Thomas & \noexpand\JEFF & >
\end{nameauth}
\PretagName[Thomas]{\noexpand\JEFF}{Jefferson, Thomas}
\TagName[Thomas]{\noexpand\JEFF}{, pres.|hyperpage}
\newcommand\JEFF{\ifDoGen\textSC{Jefferson's}\else
  \textSC{Jefferson}\fi}

Consider \Genitivetrue\Jeff\ legacy. More on \Jeff\ later.
\Genitivetrue\Jeff\ reputation has declined in recent decades.
```

Consider Thomas Jefferson's legacy. More on Jefferson later. Jefferson's reputation has declined in recent decades.

For highly inflected languages, this would require two Boolean flags per case and nested conditional statements. Now we resume normal formatting with \AltFormatInactive and we do not use the names in this section outside of it.[20]

Back to Section 1.2

[19] A copy of this example is in examples.tex, collocated with this manual.

[20] In a dtx file it is best to put the nameauth environment, \PretagName, and \TagName macros in the driver section, especially when names contain macros.

2.6 Indexing Macros

2.6.1 Index Entries and Control

\IndexName

Both package users and the naming macros themselves use this macro to create index entries. It prints nothing in the body text:

\IndexName[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

If ⟨*FNN*⟩ is present, it ignores ⟨*Alternate*⟩ for Western and "native" Eastern name forms. If ⟨*FNN*⟩ is absent, \IndexName may use the current or obsolete non-Western syntax (Section 1.4). Indexing follows [Mulvany, 152–82].

If \IndexInactive or the noindex option are used, this macro does nothing until \IndexActive appears. Additionally, it will not create index entries for cross-references made by \IndexRef and \AKA. It will not index names excluded by \ExcludeName. This provides some error protection for professional indexing.

\IndexName and \IndexRef call \@nameauth@Index, a macro that assembles an index entry from the sort tag (Section 2.6.2), name arguments, and index tag (Section 2.6.4). Different standards exist for index entries and cross-references. Check with your publisher, style guide, and docs for xindy and makeindex.

\IndexActive
\IndexInactive

The noindex option deactivates the indexing function of this package until \IndexActive enables indexing. Another macro, \IndexInactive, will deactivate indexing again. These can be used throughout the document. **\IndexInactive suppresses index sorting and tagging macros.** Compare the use of macros \ExcludeName and \IncludeName (Section 2.6.2).

\IndexProtect
3.3

Both the core name engine \@nameauth@Name and \AKA have locks that prevent them from being re-entrant. This protects the text. Usually, one does not put naming macros in the index. Just in case, now one can use \IndexProtect right before \printindex to prevent nameauth macros from producing any output.

This example shows the difference between the effects of the older and newer approaches. We use the tag § in this manual's index, but not below:

Macro	Text	.ind file	Index
no protection \Name{foo\Name{bar}} (next iteration adds) →	foo	\item foo\Name {bar} \item bar	foobar bar
\IndexProtect \Name{foo\Name{bar}} (no further output results)	foo	\item foo\Name {bar}	foo

\global

\IndexActive and \IndexInactive can be used as a pair or singly within a group. These macros override any prefix macros. \IndexProtect also can be used in a local scope. Use \global with these macros to force a global effect.

\SkipIndex
3.1

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. Since prefix macros are meant for macros that print a name, both \IndexName and \IndexRef ignore \SkipIndex and allow the Boolean flags set by the prefix macros, to "pass through" to the next naming macro. That may seem counter-intuitive.

\JustIndex
3.3

This prefix macro makes \Name, \Name*, \Fname, and the shorthands act like a one-time call to \IndexName. Flags set by the prefix macros "pass through" to the next naming macro except these three: \@nameauth@JustIndexfalse (obviously), but also \@nameauth@FullNamefalse and \@nameauth@FirstNamefalse.

- Both \AKA and \PName ignore and reset the flag set by \JustIndex.
- \SkipIndex \JustIndex \Name{A} \Name{B} is just like \JustIndex \Name{A} \SkipIndex \Name{B}. See the table on page 20.
- Version 3.3 eliminates the undocumented behavior that used to occur when not using, e.g., \JustIndex\Wash. Now any version will do:

\JustIndex\LWash \Wash	old: George Washington	new: Washington
\JustIndex\SWash \Wash	old: George	new: Washington

- The oldpass option restores the old behavior. Cf. Section 2.10.4.

Back to Section 1.2

2.6.2 Cross-References

\IndexRef
3.0

This macro emerged from the macros in Section 2.9. By default, \IndexRef creates a *see* reference from the name defined by its first three arguments to the target in its final argument:

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

The name parsing is like \IndexName, except that the final argument is neither parsed nor checked if a target entry exists. For example, to cross-reference "Sun King" with Louis XIV use: \IndexRef{Sun King}{Louis XIV}.

When \IndexRef calls \@nameauth@Index, a preexisting tag of the form ⟨*some text*⟩|⟨*some macro*⟩ is reduced to ⟨*some text*⟩. One cannot tag an extant cross-reference, but one can tag a name, then later create a *see also* reference. For related warnings activated by the verbose option, see Section 1.6.

Next we look at variant names and cross-references. Some can be handled with the ⟨*Alternate*⟩ argument. Others require more work to implement (Section 2.2.3).

- Variant names potentially can have page numbers in index entries. Cross-references cannot have page numbers.
- \DropAffix\ForgetThis\Name[J.E.]{Carter, Jr.}[Jimmy] gives a variant name: Jimmy Carter indexed under "Carter, J.E., Jr."
- \IndexRef[Jimmy]{Carter}{Carter, J.E., Jr.} makes an xref but prints nothing. We need only create this cross-reference once.
- By contrast, \AKA automatically formats the cross-reference name in the text and in the index.
- Yet \AKA has limited formatting. Instead, after creating the xref with \IndexRef, one can use \SubvertThis\Name*[Jimmy]{Carter} Jimmy Carter with full formatting, but without creating any page entries.
- \SubvertThis syncs the variant with the canonical form \DropAffix \Name*[J.E.]{Carter, Jr.}[Jimmy] Jimmy Carter. Otherwise, they would act as different names. See also Section 2.8.2.

- If we use `\Name[Jimmy]{Carter}` Carter we have to index this alternate name with the canonical one: `\IndexName[J.E.]{Carter, Jr.}`

`\SeeAlso` **3.0**

Put `\SeeAlso` before `\IndexRef`, `\AKA`, and `\PName` to make a *see also* reference for a name that has appeared already in the index. Yet one should mind the caveats:

- If `\SeeAlso\IndexRef{Bar}{Foo}` occurs on page 10, `\Name{Bar}` will not create index entries thereafter. A *see also* ref follows all page refs.
- If `\SeeAlso\IndexRef{Bar}{Foo}` occurs on page 10, `\Name{Foo}` will create index entries thereafter because it is the target of "Bar."
- If `\Name{Baz}` occurs on page 12 and `\IndexRef{Baz}{Meschugge}` on page 16, no xref will be created. A *see* reference has no page refs.

`\ExcludeName` **3.0**

This macro prevents a name from being used as either an index entry or as an index cross-reference. It will not exclude extant cross-references:

`\ExcludeName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

`\IndexRef` works best if one needs a cross-reference from a variant to the canonical name. If no cross-reference is needed, then `\ExcludeName` is used. Unlike `\IndexInactive` and `\IndexActive`, this macro works only on a per-name basis. Below we keep specific names and cross-references out of the index:

```
1 \ExcludeName[Kris]{Kringle}
2 \ExcludeName[Santa]{Claus}
3 \ExcludeName{Grinch}
4 \Name[Kris]{Kringle}                    Kris Kringle
5 \Name[Kris]{Kringle}                    Kringle
6 \AKA[Kris]{Kringle}[Santa]{Claus}       Santa Claus
```

For more examples of using `\ExcludeName` to handle variants, see Sections 2.2.3 and 2.3.4, among others. We will check on the Grinch later.

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

For those who might need to break the indexing rules set by nameauth, these two macros get the job done. They remove the protections used for exclusion and cross-referencing. These macros have the same syntax as `\ExcludeName`:

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

`\IncludeName` only removes an excluded reference created by `\ExcludeName` while `\IncludeName*` completely un-protects a cross-reference. Thereafter, one may create page entries for it like a name.

For example, we used `\ExcludeName{Attila, the Hun}` at the end of Section 1.3.2. Using `\IfAKA{Attila, the Hun}{`⟨*an xref*⟩`}{`⟨*no xref*⟩`}{`⟨*excluded*⟩`}` tells us that he is ⟨*excluded*⟩ (cf. Section 2.8.2).

Once we use `\IncludeName{Attila, the Hun}`, using `\LAttil` Attila the Hun will create a name and an index entry on this page. `\IfAKA` now tells us that he is ⟨*no xref*⟩. We again have a name that can be indexed.

Cross-references get more protection. `\IfAKA[Jay]{Rockefeller}` (a reference from Section 1.3.2) tells us that he is ⟨*an xref*⟩. If we follow the previous example and use `\IncludeName[Jay]{Rockefeller}` he still is ⟨*an xref*⟩. After using `\IncludeName*[Jay]{Rockefeller}` he finally becomes ⟨*no xref*⟩, removing all protection from that cross-reference.

combining xrefs

`\IndexRef` will not merge multiple cross-references. One must manually merge cross-references: `\IndexRef{Bar}{Baz; Foo}` makes the index entry "Bar, *see* Baz; Foo." The preferred standard (in the humanities) suggests that one avoid something like `\IndexRef{Bar}{Baz} \IndexRef{Bar}{Foo}`.

one xref many targets

There is a special case where one cross-reference can point to multiple targets, such as demonstrated in the example below:

```
1 \PretagName{\textit{Snellius}}{Snellius}
2 \IndexRef{\textit{Snellius}}{Snel van Royen, R.; Snel van Royen, W.}
3
4 Both \Name[W.]{Snel van Royen}[Willebrord] and
5 his son \Name[R.]{Snel van Royen}[Rudolph] were known
6 by the Latin moniker \Name{\textit{Snellius}}.
```

Both Willebrord Snel van Royen and his son Rudolph Snel van Royen were known by the Latin moniker *Snellius*.

location matters

`\IndexRef` prevents page numbers in cross-references, so one must plan how to set up complex cross-references. Above, `\Name{\textit{Snellius}}` produces no index entry because `\IndexRef` comes first.

Multiple connections

Below, two names are indexed with page numbers. They have *see also* cross-references to each other. One of those names also has a *see* reference to it:

- We use the canonical name to set up page references:

  `\Name{Maimonides}` . . . . . Maimonides

- Maimonides has two other names, one more used than the other. We set up his least-used name as the *see* reference:

  ```
  \IndexRef{Moses, ben-Maimon}{Maimonides}
  \Name{Moses, ben-Maimon}
  ```
  . . . . . Moses ben-Maimon

- We now have a main name with a page entry and a "*see* reference" to that name. Moses ben-Maimon has no page entries because we made the xref before we started to use the name.

- Before creating *see also* cross-references, we use the other alternate name so that all the page entries precede the cross-references:

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

- All *see also* references must come after all page references. For example, one could put both of these macros at the end of the document:

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

This space is intentionally left blank.

Continental Format Reference Work

Let us create a macro for entries in a reference work using the basic form of Continental formatting from Section 2.5.1. We enable alternate formatting, set up tags, and define an article with head-words:

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

\newcommand{\RefArticle}[4]{%
  \def\check{#2}%
  \ifx\check\empty
    \noindent\ForgetThis#1\ {#4}
  \else
    \noindent\ForceName#1\ ``\ForceName#2''
    \ForceName#3\ {#4}
  \fi
}
```

`\RefArticle` either formats the name from the first argument and appends the fourth argument, ignoring the others if the second is empty, or it formats the first three arguments and appends the fourth. We determine what those arguments mean by including specific naming macros.

```
\RefArticle%
  {\Name[Greta]{\textSC{Garbo}}}%
  {}{}%
  {(18 September 1905\,--\,15 April 1990) was a Swedish
   film actress during the 1920s and 1930s.}

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

\AltFormatInactive
```

Greta Garbo (18 September 1905 – 15 April 1990) was a Swedish film actress during the 1920s and 1930s.

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

Back to Section 1.2

2.6.3 Index Sorting

\IndexActual

The general practice for sorting with `makeindex -s` involves creating your own `.ist` file (pages 659–65 in *The Latex Companion*). The following form works with both `makeindex` and `texindy`: `\index{`⟨*sort key*⟩`@`⟨*actual*⟩`}`. By default, the "actual" character is `@`. If one needs to change the "actual" character, such as when using `gind.ist` with `.dtx` files, one would put `\IndexActual{=}` in the preamble (or driver section) before using `\PretagName`.

\PretagName
2.0

The nameauth package enables automatic index sorting using a "pretag" (see Section 1.5). `\PretagName` creates a sort key terminated with the "actual" character. Do not put the "actual" character in the "pretag":

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

One need only "pretag" names once in the preamble. Thereafter, they will be sorted automatically. For example:

```
1 \PretagName[Jan]{Łukasiewicz}{Lukasiewicz, Jan}
2 \PretagName{Æthelred, II}{Aethelred 2}
3 \PretagName[W.E.B.]{Du~Bois}{Dubois, W.E.B.}
```

Every reference to Jan Łukasiewicz, Æthelred II, and W.E.B. Du Bois is automatically tagged and sorted. One also must "pretag" names that contain spaces, macros, active characters, control spaces, non-breaking spaces, and anything that is not basic ASCII. That can differ when using `xindy` and Unicode-based LaTeX.

particles and languages

For example, the sort tag `de Soto` precedes `deal` due to the space: `de␣`. The sort tag `Desoto` falls between `derp` and `determinism`. German ä ö ü ß map to English ae oe ue ss. Yet Norwegian æ ø å follow z in that order. Check a style guide regarding collating sequences, spaces, and sorting. This is where using `xindy` can be very helpful. See also Section 2.3.4.

sub-entries

One can sort names by creating sub-entries, which depends on the index style and formatting files: `\PretagName[Some]{Name}{`⟨*category*⟩`!Name, Some}`. See the documentation for `xindy` and `makeindex`.

Below we show how `\PretagName` helps one to avoid manually sorting cross-references (cf. Section 2.3.4):

```
1 \PretagName{\textit{Doctor angelicus}}{Doctor angelicus}
2 \IndexRef{\textit{Doctor angelicus}}{Thomas, Aquinas}
3
4 Perhaps the greatest medieval theologian was
5 \Name{Thomas, Aquinas}, later known as
6 \Name{\textit{Doctor angelicus}}.
```

Perhaps the greatest medieval theologian was Thomas Aquinas, later known as *Doctor angelicus*.

`\PretagName` differs from the other tagging macros because its function is sorting entries, not appending information to entries:

- You can "pretag" any name and any cross-reference.
- You can "tag" and "untag" only page-reference names, not xrefs, but you can turn a page-reference name into a *see also* xref.
- You can undo a "tag" but you cannot undo a "pretag."

Debugging Problems with Sorting

If an entry is incorrect in the index, check the following:

- Are there any active characters, internal spaces, or control sequences in the name arguments? Use `\PretagName`.
- Is alternate formatting used consistently? Are any names used within sections of alternate formatting ever used outside of them?
- Are macros in the name arguments that can expand differently under different conditions preceded by `\noexpand`?

Since 2018 changes in the way that Unicode characters are handled in `pdflatex` and `latex` have made indexing simpler and more intuitive, e.g.

pre-2018	text		index	post-2018	text		index
	ä	→	`\IeC␣{\"a}`		ä	→	ä
	æ	→	`\IeC␣{\ae␣}`		æ	→	æ

One can test for this change and take different approaches with:

`\IfFileExists{utf8-2018.def}{`⟨*yes*⟩`}{`⟨*no*⟩`}`

One also should look at the entries in the `.idx` or `.ind` files to see how the name arguments and other index entry components are turned into index entries. If there are entries that do not work, one can find the corresponding page numbers in order to identify the problem.

Extra spaces are significant when sorting index entries, yet usually are not significant in the body text. Hidden spaces, tokens, macros, and control sequences create unique index entries that look similar, yet expand and sort differently. Some macros can add spaces to index entries. For example, index tags in this manual that include `\dag` show up as `\dag␣␣` in the index (two trailing spaces). Below we show a general form of macro that adds extra spaces to index entries:

```
1 \newcommand\Idx[1]{%
2   \protected@edef\arg{#1}%
3   \index{\arg}}
```

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

Back to Section 1.2

2.6.4 Index Tags

`\TagName` This macro creates a tag appended to all index entries for a corresponding `\Name`.
`\UntagName` The tag persists until one changes it with `\TagName` or destroys it with `\UntagName`. Tags can include life dates, regnal dates, and other information. Both `\TagName` and `\UntagName` handle their arguments like `\IndexName`:

`\TagName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*tag*⟩`}`
`\UntagName[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]`

All the indexing macros are keyed to the name patterns. `\PretagName` generates the leading sort key. `\TagName` and `\UntagName` affect the trailing content:

```
                  | \PretagName | \IndexName |
\index{Aethelred 2@|Æthelred II|, king}
                                               \TagName
                                               \UntagName
```

scholarly helps

Tags created by `\TagName` can be helpful in the indexes of academic texts by adding dates, titles, etc. `\TagName` causes the nameauth indexing macros to append "`,␣pope`" to the index entries for the popes below:

```
1 \TagName{Leo, I}{, pope}
2 \TagName{Gregory, I}{, pope}
3 Pope \Name{Leo, I} was known as \AKA{Leo, I}{Leo, the Great}.\\
4 Pope \Name{Gregory, I} was known as \Name{Gregory, I}
5 ``\ForceFN\AKA*{Gregory, I}{Gregory, the Great}.''
```

Pope Leo I was known as Leo the Great.
Pope Gregory I was known as Gregory "the Great."

`\TagName` works with all names, but not with cross-references from `\IndexRef`, `\AKA`, etc. (cf. Sections 2.6.2, 2.9). Tags also can be daggers, asterisks, and so on. For example, all fictional names in the index of this manual are tagged with §. One must add any desired spaces to the start of the tag.

same name game

We can format and index one name as two different people with `\TagName` and `\ForgetThis` (Section 2.8.1). The index tags group together their respective entries. In a normal LaTeX document one would write, e.g.:

```
1 \TagName[E.]{Humperdinck}{ (composer)}
2 This refers to the classical composer:
3 \Name[E.]{Humperdinck}[Engelbert].
4
5 \TagName[E.]{Humperdinck}{ (singer)}
6 This refers to the pop singer from the 60s and 70s:
7 \ForgetThis\Name[E.]{Humperdinck}[Engelbert].
```

This refers to the classical composer: Engelbert Humperdinck.

This refers to the pop singer from the 60s and 70s: Engelbert Humperdinck.

special tags

One can use `\TagName` to create "special" index entries for names with the general form `\TagName{`⟨*Name*⟩`}{|`⟨*Macro*⟩`}`, when `\def\`⟨*Macro*⟩`#1{#1}` exists. These tags are compatible with hyperref.[21]

3.3

For example, using the ltxdoc class with hypdoc does not create hyperlinked page entries with nameauth. This behavior does not affect normal LaTeX documents that use nameauth and hyperref. When creating this manual, we had to tag every name with: `\TagName{`⟨*Name*⟩`}{|hyperpage}` in the driver section of the `dtx` file.

In the "commented" package documentation part of a `dtx` file, the vertical bar is active. This adds an extra layer of complexity. Index tags in the documentation part must use the form: `\TagName{`⟨*Name*⟩`}{\string|hyperpage}`.

[21] Before version 3.3 these special tags did not work with hyperref. The fix was inspired by the answer of Heiko Oberdiek in: https://tex.stackexchange.com/questions/201720/index-produces-invalid-idx-entry-with-manual-style-commaparse-hyperref

Below we use the conventions of this manual to create a special tag:

```
\newcommand\Orphan[2]{#1(\hyperpage{#2})}
\TagName[Lost]{Name}{\,\S|Orphan{perdit}}
\Name[Lost]{Name}
```

Lost Name
idx file: `\indexentry{Name, Lost\,\S  |Orphan{perdit}}{`⟨*page*⟩`}`
ind file: `\item Name, Lost\,\S  \pfill \Orphan{perdit}{`⟨*page*⟩`}`

The microtype package and its `Spacing` environment may be the best solution to fix index entries and sub-entries that break badly across columns or pages. Suppose, however, that we wanted to insert manual breaks into an index at will, preferably after the final page reference in an entry.

We cannot just insert something like `\newpage`. In order to accomplish our goal, we need a helper macro that can take an argument. Below we use `\newpage`, but if we instead make use of the multicol or idxlayout packages we can replace that with `\columnbreak`. Two macros illustrate a similar concept:

```
\newcommand*{\EndBreak}[1]{#1\newpage}
\makeatletter
\newcommand*{\MidBreak}[1]{#1\newpage\@gobble}
\makeatother
```

Putting a break into the middle of an index entry is quite sketchy and probably should be avoided, but it can be done by using `\@gobble` to eat the comma after the break. Instead, breaking after the entry is preferable. That entry is a list of page numbers corresponding to several pages:

(page 10) `\Name{Some, Name}`
...
(page 15) `\Name{Some, Name}`
...
(page 18) `\TagName{Some, Name|EndBreak}%`
`\Name{Some, Name}`

If all instances of `\Name{Some, Name}` on page 18 have the same tag, there will be no duplicate page entries, hyperref will work, and the index will break after:

```
Some Name ... 10, 15, 18
```

manual entries

We can use the same macros in manual index entries. We may need to look at the `idx` or `ind` files to craft matching entries on the page that corresponds to the placement of the break:[22]

(page 18) `\SkipIndex\Name{Some, Name}%`
`\index{Some Name|EndBreak}`

Back to Section 1.2

[22] Results vary, depending on what distribution of LaTeX is being used and how old it is. As we saw in the previous section, any name with active characters needs to be handled differently before 2018 than after 2018.

2.7 "Text Tags"

\NameAddInfo

Unlike index tags, "text tags" are not printed automatically with every name managed by nameauth. Sections 2.8.2 and 2.10.2 have more examples. The macro is \long, allowing for some complexity in the ⟨*tag*⟩ argument:

\NameAddInfo[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]{⟨*tag*⟩}

For example, \NameAddInfo[George]{Washington}{(1732--99)} makes a text tag but does not print whenever \Wash "Washington" is used.

\NameQueryInfo

To print the text tag macro associated with a name, we use \NameQueryInfo, which calls the appropriate macro in the name info data set:

\NameQueryInfo[⟨*FNN*⟩]{⟨*SNN, Affix*⟩}[⟨*Alternate*⟩]

\NameQueryInfo[George]{Washington} expands to (1732–99). One can insert a space at its start or use signs like asterisks, daggers, and even footnotes, such as one for Schuyler Colfax.[23] Below is the source for footnote 23:

```
1 \NameAddInfo[Ulysses S.]{Grant}{(president from 1869 to 1877)}%
2 \NameAddInfo[Schuyler]{Colfax}{\footnote{He was the seventeenth
3 US vice-president, holding office during the first term (1869--73)
4 of \Name[Ulysses S.]{Grant} \NameQueryInfo[Ulysses S.]{Grant}.}}

  ...\Name[Schuyler]{Colfax}.\NameQueryInfo[Schuyler]{Colfax}
```

Since one can nest "text tags" or have them call each other, one can build complex relations. Yet one must protect against a stack overflow by using Boolean flags to stop the recursion:

```
1 \newif\ifA
2 \newif\ifB
3 \NameAddInfo{A}{%
4   \Atrue A \ifB Stop \else \NameQueryInfo{B} \fi \Afalse}
5 \NameAddInfo{B}{%
6   \Btrue B \ifA Stop \else \NameQueryInfo{A} \fi \Bfalse}

  \NameQueryInfo{A} → A B Stop
  \NameQueryInfo{B} → B A Stop
```

\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, Affix*⟩}[⟨*Alternate*⟩]

After using \NameClearInfo[George]{Washington}, the next attempt to query the tag \NameQueryInfo[George]{Washington} will produce nothing.[24]

Back to Section 1.2

[23] He was the seventeenth US vice-president, holding office during the first term (1869–73) of Ulysses S. Grant (president from 1869 to 1877).

[24] Had any information from a text tag been present, it would have appeared between "nothing" and the full stop.

2.8 Name Decisions

The macros in this section force and detect name states. Below we keep names consistent with beamer overlays using some of the macros explained in this section. Otherwise, name forms will change as one advances the slides:[25]

```
\documentclass{beamer}
\usepackage{nameauth}
\mode<presentation>
\beamerdefaultoverlayspecification{<+->}

\begin{document}

\begin{frame}{Move Text Without Retyping Names}
  \begin{itemize}\footnotesize
  \item<1-> Original\ForgetName[George]{Washington}%
                    \ForgetName[George]{Washington's}\\
            \Name[Martin]{Van Buren} changes
            after the first overlay.
  \begin{enumerate}
  \item<2-> \IfMainName[George]{Washington's}{He}%
            {\Name[George]{Washington}}
            became the first president
            of the United States.
  \item<3-> \IfMainName[George]{Washington}{His}%
            {\SkipIndex\Name*[George]{Washington's}}
            military successes during the Seven Years War
            readied him to command the army
            of the Continental Congress.
  \end{enumerate}
  \item<1-> Reordered\ForgetName[George]{Washington}%
                     \ForgetName[George]{Washington's}\\
            \ForgetThis\Name[Ulysses S.]{Grant}
            does not change.
  \begin{enumerate}
  \item<3-> \IfMainName[George]{Washington}{His}%
            {\SkipIndex\Name*[George]{Washington's}}
            military successes during the Seven Years War
            readied him to command the army
            of the Continental Congress.
  \item<2-> \IfMainName[George]{Washington's}{He}%
            {\Name[George]{Washington}}
            became the first president
            of the United States.
  \end{enumerate}
  \end{itemize}
\end{frame}

\end{document}
```

The overlays, numbered progressively from one to three, begin by deleting name control sequence patterns. Uncontrolled names will change. Name conditionals ensure specific, context-dependent forms based on what name has appeared. These conditionals allow the text to be order-independent.

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

2.8.1 Making Decisions

Naming system behavior

By default, the macros below produce global effects. They change both the `!MN` and `!NF` data sets (Section 1.5). With `\ForceName` (Section 2.4), they change formatting. Apart from `\ForceName`, they also change long or short name forms and the outcome of the testing macros in the next section:

Defaults	Name Length	Format Hooks	Test Path
Pre-First Use	Long	First	False
Subsequent Use	Long or short	Subsequent	True

Modifications	Form	Function
`\SubvertThis\LAnth`	Susan B. Anthony	force subsequent use force long form
`\ForceName\SAnth`	Susan B.	default subsequent use force first-use format
`\ForgetThis\SAnth`	Susan B. Anthony	force first use; default long default first-use format
`\SAnth`	Susan B.	default subsequent use default short form

`\ForgetName`

This macro takes the same arguments as `\Name`. It "forgets" a name, forcing a "pre-first use" case. The syntax is:

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

`\ForgetThis` **3.1**

This prefix macro causes the next instance of a naming macro or shorthand to "forget" a name before printing it. After knowing `\Einstein` "Einstein" we forget him and again have a first reference: `\ForgetThis\Einstein` "Albert Einstein."

`\SubvertName`

This macro takes the same arguments as `\Name`. It "subverts" a name, forcing a "subsequent use" case. The syntax is:

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

`\SubvertThis` **3.1**

This prefix macro causes the next instance of a naming macro or shorthand to "subvert" a name before printing it. As in the table on page 20, `\ForgetThis` has a higher priority than `\SubvertThis` and nullifies it when used together.

`\LocalNames` `\GlobalNames`

`\LocalNames` restricts the effects of the macros above to the current naming system. `\GlobalNames` restores the default behavior. We define a macro that reports whether a name exists in the main matter, front matter, both, or none:

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

We start with no extant name:

`\CheckChuck` . . . . . **none**

We create a name in the "main matter":

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

We switch to the "front-matter" and create a name, but since we are using the `quote` environment, we add `\global`:

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

We now have two names. Their patterns are:

```
Charlie!Chaplin!MN
Charlie!Chaplin!NF
```

We use `\Localnames` to make `\ForgetName` and `\SubvertName` work with only the front-matter system. 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 main matter, again using `\global` to ignore scoping.

```
\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 systems:

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

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

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

Back to Section 1.2

2.8.2 Testing Decisions

The macros in this section test for the presence or absence of a name, then expand based on the result. For example, they can synchronize information between a float and body text by each testing whether a name exists and making decisions about the information accordingly.

`\IfMainName` In order to test whether or not a "main matter" name control sequence exists, use this long macro that can accommodate paragraph breaks:

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

For example, because we have not seen the equivalent of `\Name[Bob]{Hope}` or `\SubvertName[Bob]{Hope}`, we try the following test and get:

`\IfMainName[Bob]{Hope}{Bob here!}{No Bob.}` . . . . . No Bob.

Still, we can create an index entry here with `\IndexName[Bob]{Hope}` and it will not affect the test above. Since, however, we have encountered the equivalent of `\Name{Elizabeth,I}` many times in the document, we get the following result:

`\IfMainName{Elizabeth,I}{Bess here!}{No Bess.}` . . . . . Bess here!

`\IfFrontName` In order to test whether or not a "front matter" name control sequence exists, use this long macro that can accommodate paragraph breaks. Its syntax is:

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

This macro works like `\IfMainName`, except using the "front matter" name control sequences as the test subject.

For example, based on Section 2.4, we see that "Carnap is both" a main-matter and front-matter name with the following test:

```
\IfFrontName[Rudolph]{Carnap}%
{%
  \IfMainName[Rudolph]{Carnap}%
    {\Name[Rudolph]{Carnap} is both}%
    {\Name[Rudolph]{Carnap} is only front-matter}%
}%
{%
  \IfMainName[Rudolph]{Carnap}%
    {\Name[Rudolph]{Carnap} is only main-matter}%
    {\Name[Rudolph]{Carnap} is not mentioned}%
}
```

`\IfAKA` This macro tests whether or not a regular or excluded form of cross-reference control sequence exists. The syntax is:

`\IfAKA[`⟨*FNN*⟩`]{`⟨*SNN, Affix*⟩`}[`⟨*Alternate*⟩`]{`⟨*y*⟩`}{`⟨*n*⟩`}{`⟨*excluded*⟩`}`

This macro works like `\IfMainName`, although it has an additional ⟨*excluded*⟩ branch in order to detect the activity of `\ExcludeName` (Section 2.6.2).

Cross-references are governed by name control sequences ending in `!PN` (Section 1.5). Regular cross-reference control sequences (the `{`⟨*y*⟩`}` path) expand to empty. Excluded control sequences (the `{`⟨*excluded*⟩`}` path) expand to `!`.

`\ExcludeName` creates excluded xrefs; `\IncludeName` destroys them. Regular xrefs are created by `\IndexRef`, `\AKA`, `\AKA*`, and `\PName`; they are destroyed by `\IncludeName*`. Here is how we use this logic:

1. In the text we refer to former pro-wrestler and Minnesota governor Jesse Ventura, `\Name[Jesse]{Ventura}`.
2. We establish his lesser-known legal name as an alias: "James Janos," `\IndexRef[James]{Janos}{Ventura, Jesse}\Name[James]{Janos}`.
3. We get the result: "Jesse Ventura is a stage name." If we do not use `\ExcludeName`, we can leave the `{`⟨*excluded*⟩`}` branch empty:

```
\IfAKA[James]{Janos}%
  {\Name*[Jesse]{Ventura} is a stage name}%
  {\Name*[Jesse]{Ventura} is a regular name}%
  {}
```

Otherwise, based on Section 2.6.2, we get: "Grinch is excluded":

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

We can combine all these macros create a complete test:

```
\IfAKA[FNN]{SNN, Affix}[Alternate]%
  {true; it is a pseudonym}%
  {% if not a pseudonym:
    \IfFrontName[FNN]{SNN, Affix}[Alternate]% yes path
    {\IfMainName[FNN]{SNN, Affix}[Alternate]%
      {both}%
      {front}%
    }%
    {\IfMainName[FNN]{SNN, Affix}[Alternate]% no path
      {main}%
      {does not exist}%
    }%
  }%
  {excluded path}
```

We can use the text tag macros with the conditional macros to present information that depends on what names have already occurred. One must avoid unbounded recursion that results in a stack overflow (Section 2.7):

```
\IndexRef{Paul}{Saul of Tarsus}
\NameAddInfo{Saul, of Tarsus}{\IfMainName{Jesus, Christ}
    {\IfMainName[Lucius]{Sergius Paulus}
      {renames himself \Name{Paul}}
      {a preacher to the Gentiles}}
    {wrote that he persecuted Christians}}

\Name{Saul, of Tarsus} \NameQueryInfo{Saul, of Tarsus}. He
saw a vision of \Name{Jesus, Christ} on the road to Damascus
and became \NameQueryInfo{Saul, of Tarsus}. After converting
\Name[Lucius]{Sergius Paulus}, \Name{Saul, of Tarsus}
\NameQueryInfo{Saul, of Tarsus} in honor of that.
```

> Saul of Tarsus wrote that he persecuted Christians. He saw a vision of Jesus Christ on the road to Damascus and became a preacher to the Gentiles. After converting Lucius Sergius Paulus, Saul renamed himself Paul in honor of that.

Using these tests inside other macros or passing control sequences to them may create false results (see *The TEXbook*, 212–15). That is why nameauth uses token registers to save name arguments (Section 2.10.2. Consider using `\noexpand` in macros passed as name arguments and see also Section 2.11.2. Using the trace package, `\show`, or `\meaning` can help one mitigate problems.

Back to Section 1.2

2.9 Alternate Name Macros

3.0 The macros in this section predate `\IndexRef` and have a syntax and behavior recalling early package versions.[26] Using `\IndexRef` with `\Name` can be more flexible (cf. page 41). To save space, we show the syntax of these macros using ⟨*SAFX*⟩ as the equivalent of ⟨*SNN, Affix*⟩. Common properties include:

- These macros do not create page references.
- The target `[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}` comes before the xref printed in the text: `[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`.
- The obsolete syntax cannot be used with `[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}`.
- Only the ⟨*SAFX*⟩ and ⟨*xref SAFX*⟩ arguments are able to use comma-delimited suffixes.
- One cannot use `\TagName` with a cross-reference, but one can sort it with `\PretagName[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}{`⟨*sort tag*⟩`}`.

`\AKA` `\AKA*` `\AKA` (*also known as*) and its starred form display an alias in the text and create a cross-reference in the index. They display and format names differently than the name macros:

```
\AKA [⟨FNN⟩]{⟨SAFX⟩}[⟨xref FNN⟩]{⟨xref SAFX⟩}[⟨xref Alternate⟩]
\AKA*[⟨FNN⟩]{⟨SAFX⟩}[⟨xref FNN⟩]{⟨xref SAFX⟩}[⟨xref Alternate⟩]
```

Both macros create a cross-reference in the index from the ⟨*xref FNN*⟩, ⟨*xref SAFX*⟩, and ⟨*xref Alternate*⟩ arguments to a target defined by ⟨*FNN*⟩ and ⟨*SAFX*⟩, regardless of whether that name exists. The order of the name and cross-reference in `\AKA` is opposite that of `\IndexRef`. Otherwise the ⟨*xref Alternate*⟩ argument would be ambiguous with ⟨*FNN*⟩. `\AKA` prints a long form of the cross-reference name in the text. `\SeeAlso` works with `\AKA`, `\AKA*`, and `\PName`.

`\AKA` prints the ⟨*xref FNN*⟩ and ⟨*xref SAFX*⟩ arguments in the body text. If ⟨*xref Alternate*⟩ is present with ⟨*xref FNN*⟩, `\AKA` swaps them in the text. If ⟨*xref Alternate*⟩ is present without ⟨*xref FNN*⟩, the old syntax is triggered, which we do not recommend. 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`.

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

`\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes Hope
`\RevComma\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Hope, Leslie Townes
`\AKA[Bob]{Hope}[Leslie Townes]{Hope}[Lester T.]`	Lester T. Hope
`\AKA*[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes
`\AKA*[Bob]{Hope}[Leslie Townes]{Hope}[Lester]`	Lester

[26] Before version 3.0 the lack of modularity resulted in separate name parsing, name display, and indexing for the naming macros and the alternate name macros. The version 3 series has corrected many early missteps while remaining compatible.

Next we have references to Louis XIV, Lao-tzu, and Gregory I, as well as Lafcadio Hearn and Charles du Fresne:

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

formatAKA

`\AKA` and its derivatives use `\MainNamesHook` and `\FrontNamesHook` to print the cross-reference because that helped keep cross-references distinct from names in early package versions.

The `formatAKA` package option allows first-use formatting of alternate names, but cross-references use their own system for being "first" (Section 1.5). We simulate `formatAKA` and use `\AKA{Elizabeth,I}[Good Queen]{Bess}`. The colors indicate which hooks are 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.

The first appearance of the cross-reference uses the first-use hooks of whatever naming system is active. Thereafter we only use the subsequent-use hooks of both systems unless we use `\ForceName`.

alwaysformat

Below we compare the behavior of the `alwaysformat` option, where all regular names use only `\NamesFormat` and `\FrontNamesFormat`:

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.

\PName
\PName*

These convenience macros (an early feature) print a main name and a cross-reference in parentheses:

`\PName [`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`
`\PName*[`⟨*FNN*⟩`]{`⟨*SAFX*⟩`}[`⟨*xref FNN*⟩`]{`⟨*xref SAFX*⟩`}[`⟨*xref Alternate*⟩`]`

The starred form `\PName*` is like the starred form `\Name*` to the extent that it prints a long form of ⟨*FNN*⟩⟨*SAFX*⟩. It does not affect the cross-reference arguments ⟨*xref FNN*⟩⟨*xref SAFX*⟩⟨*xref Alternate*⟩.

Except `\SkipIndex`, prefix macros only affect ⟨*FNN*⟩⟨*SAFX*⟩, not the cross-reference, which always takes a long form. `\SkipIndex` keeps both names out of the index. `\PName` cannot use the obsolete syntax for the main name, but it can do so for the alternate name.

Recommended Macro / Output	**Index**
`\PName[Mark]{Twain}%` `  [Samuel L.]{Clemens}`	
Mark Twain (Samuel L. Clemens)	Clemens, Samuel L. *see* Twain, Mark
Twain (Samuel L. Clemens)	Clemens, Samuel L. *see* Twain, Mark
`\PName*[Mark]{Twain}%` `  [Samuel L.]{Clemens}[Sam]`	
Mark Twain (Sam Clemens)	Clemens, Samuel L. *see* Twain, Mark
`\PName{Voltaire}%` `  [François-Marie]{Arouet}`	
Voltaire (François-Marie Arouet)	Arouet, François-Marie *see* Voltaire
Voltaire (François-Marie Arouet)	Arouet, François-Marie *see* Voltaire
`\PName{William, I}%` `  {William, the Conqueror}`	
William I (William the Conqueror)	William the Conqueror *see* William I
William (William the Conqueror)	William the Conqueror *see* William I
`\PName*{William, I}%` `  {William, the Conqueror}`	
William I (William the Conqueror)	William the Conqueror *see* William I

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

Bernard of Clairvaux (*Doctor mellifluus*)	*Doctor mellifluus see* Bernard of Clairvaux
Bernard (*Doctor mellifluus*)	*Doctor mellifluus see* Bernard of Clairvaux
`\ForgetThis\PName{Lao-tzu}{Li, Er}`	
Lao-tzu (Li Er)	Li Er *see* Lao-tzu
Lao-tzu (Li Er)	Li Er *see* Lao-tzu
Discouraged Macro / Output	**Index**
`\PName{William, I}{William}%` `  [the Conqueror]`	
William (William the Conqueror)	William the Conqueror *see* William I
`\PName{Lao-tzu}{Li}[Er]`	
Lao-tzu (Li Er)	Li Er *see* Lao-tzu

The newer non-Western syntax does not work with `\PName`. If we attempted to use `\SkipIndex\PName*{William, I}[William]{the Conqueror}`, this macro would put "William I (William the Conqueror)" in the body text, but its index entry would be incorrect: "the Conqueror, William *see* William I".

Back to Section 1.2

2.10 Longer Examples

Examples from the remainder of this manual are in `examples.tex`, included with the nameauth documentation.

dtx files

When creating package documentation, any name that has a macro in its argument should be set up in the driver section (the `nameauth` environment and tags from `\PretagName` and `\TagName`). Otherwise, errors can result.

2.10.1 Hooks: Intro

In these sections on advanced topics we reset all formatting hooks to do nothing. This helps us focus on the modifications made hereafter.

Before we get to the use of text tags and name conditionals in name formatting, we seek to illustrate that something more complex than a font switch can occur in `\NamesFormat`. Below we put the first mention of a name in boldface, along with a marginal notation if possible.

```
\let\OldFormat\NamesFormat
\renewcommand*\NamesFormat[1]{\textbf{#1}\unless\ifinner
   \marginpar{\raggedleft\scriptsize #1}\fi}
\let\NamesFormat\OldFormat
\PretagName{Vlad, Ţepeş}{Vlad Tepes} % for accented names
\TagName{Vlad, II}{ Dracul}          % for index information
\TagName{Vlad, III}{ Dracula}
```

Within the document after the preamble:

```
Name{Vlad, III}[III Dracula], known as
\AKA{Vlad III}{Vlad, Ţepeş} (the Impaler)
after his death, was the son of \Name{Vlad, II}[II Dracul],
a member of the Order of the Dragon. Later references to
``\Name*{Vlad, III}'' and ``\Name{Vlad, III}'' appear thus.
```

Vlad III Dracula
Vlad II Dracul

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

```
let\NamesFormat\OldFormat
```

Now we have reverted to the default `\NamesFormat` and we get:

- `\ForgetThis\Name{Vlad, III}[III Dracula]` . . . Vlad III Dracula
- `\Name*{Vlad, III}` . . . . . Vlad III
- `\Name{Vlad, III}` . . . . . Vlad

We also set up the cross-reference `\IndexRef{Dracula}{Vlad III}`. Compare the examples for Demetrius I in Section 2.3.4.

Back to Section 1.2

2.10.2 Hooks: Life Dates

We can use name conditionals (Section 2.8.2) and text tags (Section 2.7) to add life information to names when desired.

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

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

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

This package makes three token registers available to facilitate using the name conditional macros as we do below. These registers are necessary for names that contain accents and diacritics.[27]

Below the first use of a name is in small caps. Text tags are in boldface with naming macros, and roman with \AKA. Just because we set up a cross-reference does not mean that we use \AKA by default, as was the case in early versions of nameauth. We use \ForceName to use it more than once with \AKA:

```
\newif\ifNoTag% allows us to work around \ForgetName
\let\OldFormat\NamesFormat
\let\OldFrontFormat\FrontNamesFormat
\makeatletter
\renewcommand*\NamesFormat[1]{\begingroup%
  \protected@edef\temp{\endgroup\textsc{#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
      {\normalfont\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
```

We print tags in the first use hooks unless \NoTag is set true. This method uses the two ϵ-TEX primitives \noexpand and \unexpanded to avoid repetition of \expandafter. Since nameauth depends on etoolbox, we assume ϵ-TEX.

Before we can refer to any text tags, we must create them. Using the approach above, we include a leading space in the text tags. The leading space is needed only when a text tag appears.[28] We also set up a cross-reference, which we will use regardless of whether we also use \AKA. The cross-reference will be created only once and skipped thereafter:

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

[27] In \AKA these registers correspond to the **last** three arguments, the xref.

[28] 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: . . . `{ }\noexpand\NameQueryInfo`. . .

Now we begin with the first example, which, after all the setup, looks deceptively simple, but highly reusable without extra work:

```
\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 we already set up a cross-reference with `\IndexRef`, we can use just the the naming macros with "Atatürk" and get the desired formatting without any page references in the index:

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

First use, no tag:
\NoTagtrue\ForgetThis\Name{Atatürk}.
```

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

First use, no tag: Atatürk.

Since we set up distinct formatting for `\AKA` (`\normalfont` instead of boldface for text tags associated with cross-references), we now simulate the `formatAKA` package option and use `\ForceName` with `\AKA`:

```
\makeatletter\@nameauth@AKAFormattrue\makeatother
\ForgetThis\Name[Mustafa]{Kemal} was granted the name
\ForceName\AKA[Mustafa]{Kemal}{Atatürk}. We mention
\Name[Mustafa]{Kemal} and \AKA[Mustafa]{Kemal}{Atatürk} again.

First use, no tag:
\NoTagtrue\ForceName\AKA[Mustafa]{Kemal}{Atatürk}.
```

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

First use, no tag: Atatürk.

Please remember to reset the formatting:

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

Back to Section 1.2

2.10.3 Hooks: Advanced

Alternate Formatting

3.1 The alternate formatting framework provides features that aid both error protection and ease of use. This section uses `\AltFormatActive`. We do not use the names in this section elsewhere. A name designed for the alternate formatting regime may cause spurious index entries when used in the default formatting regime.

Both `\AltFormatActive` and `\AltFormatActive*` set the internal Boolean flag `\@nameauth@AltFormattrue`, enabling alternate formatting. `\AltFormatActive` sets `\@nameauth@DoAlttrue`, which activates formatting. `\AltFormatInactive` sets both flags false.

`\AltFormatActive*` normally suppresses formatting changes but it still forces `\CapThis` to work through `\AltCaps`. This produces the default look of `nameauth` and prevents Continental formatting, but it also reduces spurious index entries and errors if many names use macros in their arguments.

Alternate formatting protects against errors created when `\@nameauth@Cap` (used by `\CapThis`) gets a failure result from `\@nameauthUTFtest`, but that result is neither a letter nor a macro that expands to a sequence of letters. Protected macros and other cases may create errors if `\MakeUppercase` is applied to them. `\AltCaps` and `\CapThis` work together to avoid this problem (Section 2.5).

Continental Format

Here we look in greater detail at how `nameauth` implements the advanced version of Continental formatting. Font changes occur in the short macros `\textSC`, `\textIT`, `\textBF`, and `\textUC`. They all look similar to `\textSC`:

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

If the `altformat` option or `\AltFormatActive` is used, formatting occurs in both the text and in the index. We want small caps on by default in the text and index, then off in subsequent uses. Thus, we use `\AltFormatActive`, then redefine `\MainNameHook` because it is the subsequent use hook. `\AltOff` deactivates formatting only in the formatting hooks:

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

`\CapThis` now triggers `\AltCaps` to capitalize its argument:

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

We must put `\noexpand` before `\textSC`, `\AltCaps`, and so on to prevent them from expanding outside of the formatting hooks.

Before we alter the formatting hooks, we either can `\let` the hook macros to recall them later or we can use `\begingroup` and `\endgroup` to create a new scope that localizes any changes. We use scoping in this section.

This 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}
\let\FrontNameHook\MainNameHook
```

To suppress all formatting in the front-matter text, one need simply to use `\let\FrontNamesFormat\MainNameHook`. Continental formatting usually alters at least one element in the required name argument, 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. When sorting names, be sure to use `\noexpand` before the macros:

```
\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}
```

First	Next	Long	Short
`\Adams`	`\Adams`	`\LAdams`	`\SAdams`
John Adams	Adams	John Adams	John
`\SDJR`	`\SDJR`	`\LSDJR`	`\SSDJR`
Sammy Davis Jr.	Davis	Sammy Davis Jr.	Sammy
`\HAR`	`\HAR`	`\LHAR`	`\SHAR`
Harun al-Rashid	Harun	Harun al-Rashid	Harun
`\Mencius`	`\Mencius`	`\LMencius`	`\SMencius`
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, only using the macro `\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.[29]
- One must include all the macros in the name arguments.

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

[29] The way that Continental resources treat certain affixes relates to similar issues in [Mulvany, 168–73]. Handling non-Western names in Western sources can be a gray area. One ought take care to be culturally sensitive in these matters.

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

Rolling Your Own: Basic

Here we set out on the path to custom formatting by using package features that have been implemented already and look similar to the solutions in Section 2.5.

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

We recommend examining 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 91 and following. If you create your own macros, they will look similar:

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

Since `\AltCaps` is part of `nameauth`, you need not reinvent that wheel. Just use it. The final step is redefining the hooks, which can be as simple as:

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

When sorting names, be sure to use `\noexpand` as shown previously:

```
\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
`\deSmet`	`\deSmet`	`\LdeSmet`	`\SdeSmet`
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.

Some formatting, such as the use of `\textSC`, is fairly standard. Other formatting, such as `\Fbox` above, is ornamental and may be handled better with custom features (Section 2.10.4), but those features appear only in the text.

Rolling Your Own: Intermediate

"Intermediate" and "advanced" refer to the way hooks were designed before version 3.1. We begin the journey to more customized formatting by looking at `\NameParser`, whose logic Sections 3.4 and 3.6 show in detail.

`\NameParser` **3.1**

This user-accessible parser (Section 3.6) builds a printed name from the internal macros `\FNN`, `\SNN`, `\rootb` and `\suffb`. It uses the following Boolean flags:[30]

Only one or the other of these can be true to avoid undocumented behavior.

`\if@nameauth@FullName`	Print a full name if true.
`\if@nameauth@FirstName`	Print a first name if true.

Reversing without commas overrides reversing with commas.

`\if@nameauth@RevThis`	Reverse name order if true.
`\if@nameauth@EastFN`	toggled by `\ForceFN`.
`\if@nameauth@RevThisComma`	Reverse Western name, add comma.

We create a hook that can ignore ignore the output of `\@nameauth@Name`, which is the `#1` in the hook dispatcher's code `\bgroup`⟨*Hook*⟩`{#1}\egroup`:

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

With the `altformat` option or `\AltFormatActive` we can design a subsequent-use hook that deactivates formatting inside of it:

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

If we used `\AltFormatActive*`, where the formatting macros are enabled, but deactivated, then we might want a hook that activates the macros:

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

Within the hooks we can use the user-level parser as often as we want. We also can change internal Boolean flags, for example:

```
\makeatletter
\renewcommand*\NamesFormat[1]{\small%
  \hbox to 3.5em{[now]\hfill}\space\NameParser\\%
  \@nameauth@FullNametrue%
  \hbox to 3.5em{[long]\hfill}\space\NameParser\\%
  \@nameauth@FullNamefalse%
  \@nameauth@FirstNametrue%
  \hbox to 3.5em{[short]\hfill}\space\NameParser}
\makeatother
\let\MainNameHook\NamesFormat
```

`\JRIV` displays:

[now] Rockefeller
[long] John David Rockefeller IV
[short] John David

The proof of concept above is interesting, but not very useful. Now we move on toward more useful designs, based on Sections 2.10.1 and 2.10.2.

[30] These exclude all capitalization macros.

We begin by defining a name that is composed only of macros:

```
\begin{nameauth}
  \< Shak & \noexpand\WM & \noexpand\SHK & >
\end{nameauth}
\PretagName[\noexpand\WM]{\noexpand\SHK}{Shakespeare, William}
\PretagName[Robert]{\textSC{Burns}}{Burns, Robert}
```

Now we define the flags by which the macros `\WM` and `\SHK` expand differently in the formatting hooks than in the index:

```
\newif\ifSpecialFN
\newif\ifSpecialSN
\newif\ifRevertSN
\newcommand*\WM{\ifSpecialFN Wm.\else William\fi}
\newcommand*\SHK{\ifRevertSN \textSC{Shakespeare}\else
                 \ifSpecialSN \noexpand\AltCaps{t}he Bard\else
                 \textSC{Shakespeare}\fi\fi}
\newcommand*\Revert{\RevertSNtrue}
\makeatletter
```

Finally, we define the two formatting hooks that trigger these changes:

```
\renewcommand*\NamesFormat[1]{%
  \RevertSNfalse\SpecialFNfalse\SpecialSNfalse#1%
  \unless\ifinner\marginpar{%
    \footnotesize\raggedleft%
    \@nameauth@FullNametrue%
    \@nameauth@FirstNamefalse%
    \@nameauth@EastFNfalse%
    \SpecialFNtrue\SpecialSNfalse%
    \NameParser}%
  \fi\global\RevertSNfalse}
\renewcommand*\MainNameHook[1]{%
  \AltOff\SpecialFNfalse\SpecialSNtrue#1%
  \unless\ifinner
    \unless\ifRevertSN
      \marginpar{%
      \footnotesize\raggedleft%
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
      \@nameauth@EastFNfalse%
      \SpecialFNfalse\SpecialSNfalse%
      \NameParser}%
    \fi
  \fi\global\RevertSNfalse}
\makeatother
```

Wm. Shakespeare
Robert Burns

Shakespeare

William Shakespeare `\Shak` is the national poet of England in much the same way as Robert Burns `\Name[Robert]{\textSC{Burns}}` is that of Scotland. With the latter's rise of influence in the 1800s, Shakespeare `\Revert\Shak` became known as "the Bard" `\Shak`.

First, we put macros `\WM` and `\SHK` in name arguments using `\noexpand`. That will make the index work properly. We use `\PretagName` to sort the names. We set up three flags. One is for `\WM` and two are for `\SHK`. `\Revert` is used to print a last name without a margin note.

In the first-use hook we allow only the canonical name via `\RevertSNfalse`, `\SpecialFNfalse`, and `\SpecialSNfalse`. The default global formatting state is set by `\AltFormatActive`. We print the canonical name in the body text. If not in inner horizontal mode, we print a margin paragraph with an alternate full name using `\NameParser` and two flags. Both hooks set `\RevertSNfalse` so that `\Revert` works on a per-name basis. The subsequent-use hook disables formatting with `\AltOff`, but it allows variant forms.

Rolling Your Own: Advanced

Here is how formatting hooks were designed before version 3.0. Updating these older hooks is helpful, but may not be necessary. Here we do not use the internal package macros. We only use `\NameParser` in the hooks to produce output. We still recommend using `\AltFormatActive` to prevent problems with `\CapThis`.

Three flags replace package internals. `\@nameauth@DoAlt` activates formatting; `\@nameauth@DoCaps` is set by `\CapThis`; and `\@nameauth@InHook` is set by the hook dispatcher. Setting `\Fboxtrue` is equivalent to using `\AltFormatActive`:

```
\newif\ifFbox% Replaces \@nameauth@DoAlt
\newif\ifFirstCap% Replaces \@nameauth@DoCaps
\newif\ifInHook% Replaces \@nameauth@InHook
\Fboxtrue
```

The formatting macro is like what we have seen, except it refers to `\ifFbox`:

```
\renewcommand*\Fbox[1]{%
  \ifFbox\protect\fbox{#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 reproduce the logic and macros that the package would have provided. That means defining everything, including `\NamesFormat`, from scratch:

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

Changes to `\ifInHook` (default false) and `\ifFbox` (default true) are local to the scope in which the hook macros are called. `\ifFirstCap` must be set globally. Below we reproduce the logic of `\AltOff` before `\NameParser`:

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

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 following names under the same entry as the "new-style" macros.

First	Next	Long	Short
`\deSmet`	`\deSmet`	`\LdeSmet`	`\SdeSmet`
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, shown below in abbreviated fashion. We keep the flags `\ifFirstCap` and `\ifInHook`. We also keep the redefined `\AltCaps`, `\CapThis`, and `\NamesFormat`. We then add:

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

- 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, only using the macro `\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.

We now resume normal formatting with `\AltFormatInactive` and close the scope that we began at the start of Section 2.10.1.

Back to Section 1.2

2.10.4 Customization

Assuming that redefining hooks and adding control sequences is insufficient, one could redesign the core name macros partially or wholly, then hook those modifications into the nameauth package without needing to patch the style file itself.

\NameauthName
\NameauthLName
\NameauthFName

All these macros are set by default to \@nameauth@Name, the internal name parser. \Name, or an unmodified shorthand, calls \NameauthName. \Name*, or an L-shorthand, sets \@nameauth@FullNametrue, then calls \NameauthLName. \FName, or an S-shorthand, sets \@nameauth@FirstNametrue, then calls \NameauthFName. One should not modify \Name and \FName directly.

Next we see a minimal working example that implements the obsolete syntax. We use few internal Boolean values, save those governing name forms. We do not implement short forms or any other features in nameauth. We must index the names with \IndexName. This example shows how to hook these redefined macros into the user interface:

```
\makeatletter
\newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
  \protected@edef\a{\trim@spaces{#1}}%
  \protected@edef\b{\trim@spaces{#2}}%
  \protected@edef\c{\trim@spaces{#3}}%
  \ifx\b\empty fail \else
    \ifx\a\empty
      \ifx\c\empty \hbox to 5em{Mononym:\hfill} {\b}\else
      \hbox to 5em{Eastern:\hfill} {\b\ \c}\fi
    \else
      \ifx\c\empty \hbox to 5em{Western:\hfill} {\a\ \b}\else
      \hbox to 5em{Alternate:\hfill} {\c\ \b}\fi
    \fi
  \fi
  \global\@nameauth@FullNamefalse%
  \global\@nameauth@FirstNamefalse%
}
\makeatother
\let\MyLName\MyName
\let\MyFName\MyName
\renewcommand*\NameauthName{\MyName}
\renewcommand*\NameauthLName{\MyLName}
\renewcommand*\NameauthFName{\MyFName}
\IndexName[George]{Washington}
\IndexName[M.T.]{Cicero}
\IndexName{Dagobert}[I]
\IndexName{Aristotle}
```

\Wash	Western:	George Washington
\Cicero[Marcus Tullius]	Alternate:	Marcus Tullius Cicero
\Dagb	Eastern:	Dagobert I
\Aris	Mononym:	Aristotle

The previous example is not particularly useful. There is, however, a more practical use for these macros. One could choose to implement additional features, then pass the information in the name argument token registers to the extant parsing macros of nameauth (cf. Section 2.10.2).

Below we introduce formatting that is additional to, inter-operative with, yet distinct from the formatting hooks:

```
\makeatletter
\newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
  \@nameauth@toksa\expandafter{#1}%
  \@nameauth@toksb\expandafter{#2}%
  \@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Normal: \hfill}%
  \fcolorbox{black}{gray!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\newcommandx*\MyLName[3][1=\@empty, 3=\@empty]{%
  \@nameauth@toksa\expandafter{#1}%
  \@nameauth@toksb\expandafter{#2}%
  \@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Long: \hfill}%
  \fcolorbox{black}{green!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\newcommandx*\MyFName[3][1=\@empty, 3=\@empty]{%
  \@nameauth@toksa\expandafter{#1}%
  \@nameauth@toksb\expandafter{#2}%
  \@nameauth@toksc\expandafter{#3}%
  \hbox to 4em{Short: \hfill}%
  \fcolorbox{black}{yellow!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
}
\makeatother
\renewcommand*\NamesFormat[1]
  {\hbox to 9em{\hfil\scshape#1\hfil}}
\renewcommand*\MainNameHook[1]{\hbox to 9em{\hfil#1\hfil}}
\renewcommand*\NameauthName{\MyName}
\renewcommand*\NameauthLName{\MyLName}
\renewcommand*\NameauthFName{\MyFName}

\ForgetName[Adolf]{Harnack}
```

\Harnack	Normal:	Adolf Harnack
\LHarnack[Adolf von]	Long:	Adolf von Harnack
\Harnack	Normal:	Harnack
\SHarnack	Short:	Adolf

3.3 After the name is printed in the body text, the internal macros **globally** set `\@nameauth@FullNamefalse` and `\@nameauth@FirstNamefalse`, as well as other flags related to the prefix macros. This prevents certain cases of undocumented behavior in versions of `nameauth` before 3.3, where resetting flags locally could cause unexpected name forms. If an existing document leverages the local resetting of flags, one can use the `oldreset` option. Compare Section 2.6.1.

`\global` Like many of the 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`.

Back to Section 1.2

2.11 Technical Notes

This manual was created with **pdflatex**

Thanks

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

2.11.1 General

About the package itself:

- For version 3.2 behavior, use both the `oldpass` and `oldreset` options.
- For version 2.6 behavior, use `oldpass`, `oldreset`, and `oldAKA`.
- The package works with both `xindy` and `makeindex`.

3.0 • Name output, index entry creation, and index cross-reference creation occur in independent modules.

3.0 • Use the `verbose` option for warnings about indexing.

- The `nameauth` environment always will 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 1.4, 2.3.1, and 2.9.

2.5 • No formatting is selected by default.

About the manual:

- This manual is the test suite.
- This manual is designed for both current and older LaTeX distributions.

3.3 • This manual has been redesigned.

- It is compatible with both A4 and US letter formats.
- We mention when this manual changes package internals.

About package building:

- The nameauth package requires etoolbox, suffix, trimspaces, and xargs. The `dtx` file encoding is UTF-8; we assume Unicode support.
- We tested this release in `dvi` mode (`latex` and `dvilualatex`), and in `pdf` mode (`pdflatex`, `lualatex`, and `xelatex`). We used `makeindex`.
- This release has been tested on GNU/Linux (distro TL 2017 and vanilla TL 2019), and Windows (MikTeX, using GNU make via Cygwin.)
- The release uploaded to CTAN is generated using `pdflatex` in GNU/Linux.

2.11.2 Active Unicode

With `\usepackage[T1]{fontenc}` we can use many active Unicode characters automatically.[31] We already covered using `\PretagName` to sort names with these characters (Section 2.6.3). Below we group by accents and diacritical marks:

acute	Á Ć É Ǵ • Í Ĺ Ń Ó Ŕ Ś Ú Ý Ź	á ć é ǵ • í ĺ ń ó ŕ ś ú ý ź
grave	À • È • • Ì Ò Ù	à • è • • ì ò ù
circumflex	Â Ĉ Ê Ĝ Ĥ Î Ĵ Ô Ŝ Û Ŵ Ŷ	â ĉ ê ĝ ĥ î ĵ ô ŝ û ŵ ŷ
tilde	Ã • • • • Ĩ Ñ Õ Ũ	ã • • • • ĩ ñ õ ũ
diaresis[32]	Ä • Ë • • Ï Ö Ü Ÿ	ä • ë • • ï ö ü ÿ
cedilla	• Ç • Ģ Ķ Ļ Ņ Ŗ Ş Ţ	• ç • ģ ķ ļ ņ ŗ ş ţ
macron	Ā • Ē Ḡ • Ī Ō Ū Ǣ Ȳ	ā • ē ḡ • ī ō ū ǣ ȳ
breve	Ă • • Ğ • Ĭ Ŏ Ŭ	ă • • ğ • ĭ ŏ ŭ
dot / dotless	Ḃ Ċ Ė Ġ • İ Ż	ḃ ċ ė ġ • ı ż
ogonek	Ą • Ę • • Į Ǫ Ų	ą • ę • • į ǫ ų
caron	Ǎ Č Ď Ě Ǧ Ǐ Ǩ Ľ Ň Ǒ Ř Š Ť Ǔ Ž	ǎ č ď ě ǧ ǐ j ǩ ľ ň ǒ ř š ť ǔ ž
various	Å Æ Đ (eth) Đ (stroke) IJ Ł Ŋ	å æ ð đ ij ł ŋ
	Ø Œ Ő Ů Ű Ș Ț Þ	ø œ ő ů ű ș ß ț þ

Additional Unicode characters can be made available when using fonts with TS1 glyphs (pages 455–463 in *The Latex Companion*). Compare the list: `http://tug.ctan.org/info/symbols/comprehensive/` or `texdoc comprehensive`.

When using a font with TS1 glyphs and slots, the following preamble snippet lets one add more Unicode characters. That enables one to write, "In Congreſs, July 4, 1776" as ``In Congreſs, July 4, 1776'':

```
\usepackage[utf8]{inputenc} % For older TL releases
\usepackage[TS1,T1]{fontenc}
\usepackage{lmodern}% Contains TS1 glyph 115
\usepackage{newunicodechar}
\DeclareTextSymbolDefault{\textlongs}{TS1}
\DeclareTextSymbol{\textlongs}{TS1}{115}
\newunicodechar{ſ}{\textlongs}
```

Many Unicode characters have native support in `xelatex` and `lualatex`, but not in `pdflatex`. Yet the latter has certain features (e.g., with respect to microtype) that others lack. The features of `makeindex` do not always equate to those in `xindy`. Those differences impact design choices.

Before 2018, some index styles excluded characters with macrons, e.g., ā. Even now, control sequences like `\=a` in the index create undocumented behavior when using `makeindex` and `gind.ist`, which changes the "actual" character from `@` to `=`. Since 2018, names like Ghazāli work properly due to new Unicode conventions. We allow for backward compatibility using the iftex package thus:

[31] As of release, most documents typeset with `latex` and `pdflatex` do not require explicit loading of either inputenc or inputenx.

[32] A diaresis mark is one way to indicate an umlaut, a sound change. German originally used a superscript `e` over `a`, `o`, and `u`. The cursive form of `e` simplified to a diaresis mark in the 1800s. A diaresis mark also signals a diaresis: reading a diphthong as two monophthongs.

```
\ifPDFTeX
  \IfFileExists{utf8-2018.def}%
    {\Name{Ghazāli}}{\Name{Ghazali}}%
  \else\Name{Ghazāli}%
\fi
```

TEX macros that partition their arguments can break active Unicode characters. Consider the simple macro `\def\foo#1#2#3!{<#1#2><#3>}`. It takes three undelimited arguments and groups the first two, then the third:

Argument	Macro	Engine	Result
`abc`	`\foo abc!`	(any)	<ab><c>
`{æ}bc`	`\foo {æ}bc!`	(any)	<æb><c>
`\ae bc`	`\foo \ae bc!`	(any)	<æb><c>
`æbc`	`\foo æbc!`	`xelatex`	<æb><c>
`æbc`	`\foo æbc!`	`lualatex`	<æb><c>
`æbc`	`\foo æbc!`	`pdflatex`	<æ><bc>
`æbc`	`\foo æbc!`	`latex`	<æ><bc>

The letter `a` is one argument. Since `{æ}` is in a group, it is one argument. The macro `\ae` also is one argument. Thus, the first two glyphs are grouped together in `#1#2` and `c` is left by itself in `#3`. Both `xelatex` and `lualatex` likewise treat the Unicode letter `æ` as one argument.

In `latex` and `pdflatex`, however, `æ` is an active Unicode control sequence that uses two arguments: `#1#2`. The tail of the input, `bc`, is crowded into `#3`. Any macro where this `#1#2` pair is divided into `#1` and `#2` will produce one of two errors: `Unicode char ...not set up for LaTeX` or `Argument of \UTFviii@two@octets has an extra }`.

3.0 We test if `\Umathchar` is not defined. If so, we check if the leading token of the argument matches the start of an active Unicode control sequence: If `\@car`⟨*test*⟩`\@nil` is equal to `\@car ß\@nil` (page 78) we capitalize `#1#2`, otherwise just `#1`. Should `#1` be a protected macro or something that does not expand to a sequence of letters, we use alternate formatting and `\AltCaps` (Section 2.5.2).

A macro defined like `\edef\foo{\CapThis\Name{bar}}` will fail. However, `\CapThis\Name{bar}` can be an argument to a macro defined with `\edef` or `\xdef`.

LATEX removes spaces between undelimited macro arguments, except the trailing argument. We use `\trim@spaces` to address this in `nameauth`. Explicit spacing macros change the results, but also require sorting with `\PretagName`. See also Sections 1.6 and 2.3.4, as well as Section 2.6.3.

Back to Section 1.2

2.11.3 LaTeX Engines

The following preamble snippet lets one build nameauth also with older TL versions. We do not load `iftex.sty` if it does not exist. We load the transitional packages when iftex is absent or older than 2019:[33]

```
\IfFileExists{iftex.sty}{\usepackage{iftex}}{}
\unless\ifdefined\RequireTUTeX
  \usepackage{ifxetex}
  \usepackage{ifluatex}
  \usepackage{ifpdf}
\fi
```

Next we test for the LaTeX engine and include packages accordingly. We could just include inputenc either way, but we are illustrating a point about testing. Some statements below should be modified, depending on one's workflow.

```
\newif\ifDoTikZ                         % Perhaps not needed
\ifxetex
  \usepackage{fontspec}
  \usepackage{polyglossia}
  \setdefaultlanguage{american}         % Use own language
  \usepackage{tikz}
  \DoTikZtrue                           % Perhaps not needed
\else
  \ifluatex
    \ifpdf
      \usepackage{fontspec}
      \usepackage{polyglossia}
      \setdefaultlanguage{american}     % Use own language
      \usepackage{tikz}
      \DoTikZtrue                       % Perhaps not needed
    \else
      \IfFileExists{utf8-2018.def}{}
      {\usepackage[utf8]{inputenc}}
      \usepackage[TS1,T1]{fontenc}
      \usepackage[american]{babel}      % Use own language
      \usepackage{lmodern}
      % Perhaps add \usepackage{tikz}
    \fi
  \else
    \IfFileExists{utf8-2018.def}{}
    {\usepackage[utf8]{inputenc}}
    \usepackage[TS1,T1]{fontenc}
    \usepackage[american]{babel}        % Use own language
    \usepackage{lmodern}
    \ifpdf                              % Perhaps not needed
      \usepackage{tikz}
      \DoTikZtrue                       % Perhaps not needed
    \fi
  \fi
\fi
```

[33] A copy of this example is in `examples.tex`, collocated with this manual.

For the sake of comparing `dvi` viewers `xdvi`, `yap`, and others, we load tikz only when making a `pdf` because some `dvi` viewers crash otherwise. This may be wholly unnecessary in a `dvips` workflow or the like. With fontspec, Latin Modern is the default. If we only make `pdf` documents, the test simplifies to testing for `\Umathchar`, then loading either fontspec (success) or fontenc (failure).

In the body text we can use something like the test below for doing `pdf` things

```
\ifDoTikZ
  doing \texttt{pdf} things\else
  doing \texttt{dvi} things\fi
```

The following equivalent conditional statements can help a macro or just the body text to work under multiple engines:

```
\ifxetex xelatex%
\else
  \ifluatex
    \ifpdf lualatex (pdf)%
    \else lualatex (dvi)%
    \fi
  \else
    \ifpdf pdflatex%
    \else latex (dvi)%
    \fi
  \fi
\fi
```

```
\unless\ifxetex
  \unless\ifluatex
    \ifpdf pdflatex%
    \else latex (dvi)%
    \fi
  \else
    \ifpdf lualatex (pdf)%
    \else lualatex (dvi)%
    \fi
  \fi
\else xelatex%
\fi
```

Back to Section 1.2

This space is intentionally left blank.

3 Implementation

3.1 Flags and Registers

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

Who Called Me?

Various macros use these flags to protect against stack overflows or choose the right output.

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

Core Macro Locks

The macros `\@nameauth@Name` and `\AKA`, with some auxiliary macros, process names in a "locked" state to avoid a stack overflow. The `BigLock` always locks the macros, preventing execution. See also Sections 2.10.2 and 2.10.3.

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

Indexing

The indexing flags permit or prevent indexing and tags. `\IndexActive` and `\IndexInctive` 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`:

```
\newif\if@nameauth@DoIndex
\newif\if@nameauth@SkipIndex
\newif\if@nameauth@JustIndex
```

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

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

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

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

Formatting

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

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

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

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

These two flags are used only for backward compatibility. The first broadly determines how per-name flags are reset, while the second affects the behavior of `\JustIndex`.

```
\newif\if@nameauth@OldReset
\newif\if@nameauth@OldPass
```

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

```
\newif\if@nameauth@Forget
\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.

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

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

Affix Commas

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

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

Name Breaking

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

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

```
\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. The default is to reset both globally on a per-name basis.

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

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

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

Eastern Names

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

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

Last-Comma-First

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

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

Cap First Letter and Format

The next flags deal with first-letter capitalization. `\CapThis` sets the first Boolean value. The second is triggered by `\@nameauth@UTFtest` when it encounters an active Unicode character. The third is a fallback triggered by `\AccentCapThis`. The fourth disables `\CapThis` for alternate formatting. The fifth toggles alternate formatting.

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

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

Name Argument Token Registers

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

These three token registers contain the current values of the name arguments passed to `\Name`, its variants, and the cross-reference arguments of `\AKA`. Users can access them especially in formatting hooks.

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

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

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

3.2 Hooks

`\NamesFormat` Post-process "first" instance of final complete name form in text. See Sections 2.4 and 2.10.1f. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

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

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 2.4 and 2.10.1f. Called when `\@nameauth@MainFormat` is true and the Boolean flag `\@nameauth@FirstFormat` is false.

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

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

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

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

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

`\NameauthName` The last three hooks usually point to `\@nameauth@Name`. See Section 2.10.4.

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

`\NameauthLName` Customization hook called after `\@nameauth@FullName` is set true. See Section 2.10.4.

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section 2.10.4.

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

3.3 Package Options

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

```
\DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
\DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
\DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
\DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
\DeclareOption{oldreset}{\@nameauth@OldResettrue}
\DeclareOption{oldpass}{\@nameauth@OldPasstrue}
\DeclareOption{index}{\@nameauth@DoIndextrue}
\DeclareOption{noindex}{\@nameauth@DoIndexfalse}
\DeclareOption{pretag}{\@nameauth@Pretagtrue}
\DeclareOption{nopretag}{\@nameauth@Pretagfalse}
\DeclareOption{allcaps}{\@nameauth@AllCapstrue}
\DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
\DeclareOption{allreversed}%
  {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
\DeclareOption{allrevcomma}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommatrue}
\DeclareOption{notreversed}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
\DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
\DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
\DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
\DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
\DeclareOption{verbose}{\@nameauth@Verbosetrue}
\DeclareOption{altformat}{%
  \@nameauth@AltFormattrue\@nameauth@DoAlttrue}
\ExecuteOptions%
  {nocomma,mainmatter,index,pretag,%
   normalcaps,notreversed,noformat}
\ProcessOptions\relax
```

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

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

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

This space is intentionally left blank.

3.4 Internal Macros

Name Control Sequence: Who Am I?

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

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

Parsing: Root and Suffix

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

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

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

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

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

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

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

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

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

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

\@nameauth@@Suffix Throw out the radix; return the suffix with no leading spaces. We use this when printing the suffix.

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

\@nameauth@GetSuff The following two macros just grab the suffix for testing if the first non-space character is an active character from inputenc.

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

\@nameauth@@GetSuff Throw out the radix; return the suffix.

```
\def\@nameauth@@GetSuff#1,#2,#3\\{#2}
```

Parsing: Capitalization

\@nameauth@TestToks Test if the leading token is the same as the leading token of an active Unicode character, using an *Esszett* (ß) as the control. We only run this macro if we are in the inputenc regime.

```
\newcommand*\@nameauth@TestToks[1]
{%
  \toks@\expandafter{\@car#1\@nil}%
  \edef\one{\the\toks@}%
  \toks@\expandafter{\@carß\@nil}%
  \edef\two{\the\toks@}%
  \ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi
}
```

\@nameauth@UTFtest Before we attempt at capitalizing anything, we need to determine if we are running under `xelatex` or `lualatex` by testing for `\Umathchar`. Then we see if inputenc is loaded. We set up the comparison and pass off to `\@nameauth@TestToks`.

```
\newcommand*\@nameauth@UTFtest[1]
{%
  \def\testarg{#1}%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \if@nameauth@Accent
        \@nameauth@UTFtrue\@nameauth@Accentfalse%
      \else
        \expandafter\@nameauth@TestToks\expandafter{\testarg}%
      \fi
    \else
      \@nameauth@UTFfalse%
    \fi
  \fi
}
```

\@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`.

```
\newcommand*\@nameauth@UTFtestS[1]
{%
  \let\ex\expandafter%
  \ex\def\ex\testarg\ex{\@nameauth@GetSuff{#1}}%
  \ex\toks@\ex\ex\ex{\testarg}%
  \ex\def\ex\test@rg\ex{\the\toks@}%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \if@nameauth@Accent
        \@nameauth@UTFtrue\@nameauth@Accentfalse%
      \else
        \expandafter\@nameauth@TestToks\expandafter{\test@rg}%
      \fi
    \else
      \@nameauth@UTFfalse%
    \fi
  \fi
}
```

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

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

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

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

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

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

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

```
\def\@nameauth@C@pUTF#1#2#3\\%
  {\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.

```
\newcommand*\@nameauth@TestDot[1]
{%
  \def\TestDot##1.\TestEnd##2\\{\TestPunct{##2}}%
  \def\TestPunct##1{%
    \ifx\TestPunct##1\TestPunct%
    \else
      \@nameauth@Puncttrue%
    \fi
  }%
  \@nameauth@Punctfalse%
  \TestDot#1\TestEnd.\TestEnd\\%
}
```

`\@nameauth@CheckDot` We assume that `\expandafter` precedes the invocation of `\@nameauth@CheckDot`, which only is called if the terminal character of the input is a period. We evaluate the lookahead `\@token` while keeping it on the list of input tokens.

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

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

```
\newcommand*\@nameauth@EvalDot%
{%
  \let\@period=.%
  \ifx\@token\@period\expandafter\@gobble \fi
}
```

Error Detection

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

```
\newcommand*\@nameauth@Error[2]
{%
  \edef\msga{#2 SNN arg empty}%
  \edef\msgb{#2 SNN arg malformed}%
  \protected@edef\testname{\trim@spaces{#1}}%
  \protected@edef\testroot{\@nameauth@Root{#1}}%
  \ifx\testname\@empty
    \PackageError{nameauth}{\msga}%
  \fi
  \ifx\testroot\@empty
      \PackageError{nameauth}{\msgb}%
  \fi
}
```

Core Name Engine

\@nameauth@Name

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

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

Both \@nameauth@Name and \AKA engage the lock below, preventing a stack overflow. Tell the formatting mechanism that it is being called from \@nameauth@Name.

```
  \if@nameauth@BigLock\@nameauth@Locktrue\fi
  \unless\if@nameauth@Lock
    \@nameauth@Locktrue%
    \@nameauth@InNametrue%
```

Test for malformed input.

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

If we use \JustIndex then skip everything else. The oldpass option restores what we did before version 3.3, where we locally reset \@nameauth@JustIndexfalse and were done. Now, however, the default is a global reset to avoid undocumented behavior.

```
    \if@nameauth@JustIndex
      \IndexName[#1]{#2}[#3]%
      \if@nameauth@OldPass
        \@nameauth@JustIndexfalse%
      \else
        \if@nameauth@OldReset
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
          \@nameauth@JustIndexfalse%
        \else
          \global\@nameauth@FullNamefalse%
          \global\@nameauth@FirstNamefalse%
          \global\@nameauth@JustIndexfalse%
        \fi
      \fi
    \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. Wrap the name with two index entries in case a page break occurs between name elements.

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

Reset all the “per name” Boolean values. The default is global.

```
      \if@nameauth@OldReset
        \@nameauth@SkipIndexfalse%
```

```
      \@nameauth@Forgetfalse%
      \@nameauth@Subvertfalse%
      \@nameauth@NBSPfalse%
      \@nameauth@NBSPXfalse%
      \@nameauth@DoCapsfalse%
      \@nameauth@Accentfalse%
      \@nameauth@AllThisfalse%
      \@nameauth@ShowCommafalse%
      \@nameauth@NoCommafalse%
      \@nameauth@RevThisfalse%
      \@nameauth@RevThisCommafalse%
      \@nameauth@ShortSNNfalse%
      \@nameauth@EastFNfalse%
    \else
      \global\@nameauth@SkipIndexfalse%
      \global\@nameauth@Forgetfalse%
      \global\@nameauth@Subvertfalse%
      \global\@nameauth@NBSPfalse%
      \global\@nameauth@NBSPXfalse%
      \global\@nameauth@DoCapsfalse%
      \global\@nameauth@Accentfalse%
      \global\@nameauth@AllThisfalse%
      \global\@nameauth@ShowCommafalse%
      \global\@nameauth@NoCommafalse%
      \global\@nameauth@RevThisfalse%
      \global\@nameauth@RevThisCommafalse%
      \global\@nameauth@ShortSNNfalse%
      \global\@nameauth@EastFNfalse%
    \fi
  \fi
  \@nameauth@Lockfalse%
  \@nameauth@InNamefalse%
```

Close the "locked" branch.

```
  \fi
```

Call the full stop detection.

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

`\@nameauth@Parse` Parse and print a name in the text. The final required argument tells us which naming system we are in (Section 1.5). Both `\@nameauth@Name` and `\AKA` call this parser.

```
\newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
{%
  \if@nameauth@BigLock\@nameauth@Lockfalse\fi
  \if@nameauth@Lock
    \let\ex\expandafter%
```

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

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

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

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

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

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

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

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

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

```
\if@nameauth@DoCaps
  \let\carga\arga%
  \let\crootb\rootb%
  \let\csuffb\suffb%
  \let\cargc\argc%
  \unless\if@nameauth@AltFormat
```

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

```
    \unless\ifx\arga\@empty
      \def\test{#1}%
      \ex\@nameauth@UTFtest\ex{\test}%
      \if@nameauth@UTF
        \ex\def\ex\carga\ex{\ex\@nameauth@CapUTF\ex{\test}}%
      \else
        \ex\def\ex\carga\ex{\ex\@nameauth@Cap\ex{\test}}%
      \fi
    \fi
```

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

```
    \def\test{#2}%
    \ex\@nameauth@UTFtest\ex{\test}%
    \if@nameauth@UTF
      \ex\def\ex\crootb\ex{\ex\@nameauth@CapUTF\ex{\rootb}}%
    \else
      \ex\def\ex\crootb\ex{\ex\@nameauth@Cap\ex{\rootb}}%
    \fi
```

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

```
    \unless\ifx\suffb\@empty
      \def\test{#2}%
      \ex\@nameauth@UTFtestS\ex{\test}%
      \protected@edef\test{\@nameauth@GetSuff{#2}}%
      \if@nameauth@UTF
        \protected@edef\test{\@nameauth@Suffix{#2}}%
        \ex\def\ex\csuffb\ex{\ex\@nameauth@CapUTF\ex{\test}}%
      \else
        \edef\test{\@nameauth@Suffix{#2}}%
        \ex\def\ex\csuffb\ex{\ex\@nameauth@Cap\ex{\test}}%
      \fi
    \fi
```

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

```
    \unless\ifx\argc\@empty
```

```
          \def\test{#3}%
          \ex\@nameauth@UTFtest\ex{\test}%
          \if@nameauth@UTF
            \ex\def\ex\cargc\ex{\ex\@nameauth@CapUTF\ex{\test}}%
          \else
            \ex\def\ex\cargc\ex{\ex\@nameauth@Cap\ex{\test}}%
          \fi
        \fi
      \fi
      \let\arga\carga%
      \let\rootb\crootb%
      \let\suffb\csuffb%
      \let\argc\cargc%
    \fi
```

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

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

Use non-breaking spaces and commas as desired.

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

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.

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

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

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

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

```
        \ifx\suffb\@empty
          \let\FNN\argc%
          \let\SNN\rootb%
```

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

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

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

`\@nameauth@NonWest` Print non-Western names from `\@nameauth@name` and `\AKA`. We inherit internal macros from the parser and do nothing apart from the locked state.

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

```
        \if@nameauth@RevThis
          \@nameauth@Hook{\FNN\Space\SNN}%
        \else
          \@nameauth@Hook{\SNN\Space\FNN}%
        \fi
      \else
        \if@nameauth@FirstName
          \if@nameauth@EastFN
            \@nameauth@Hook{\FNN}%
          \else
            \@nameauth@Hook{\SNN}%
          \fi
        \else
          \@nameauth@Hook{\SNN}%
        \fi
      \fi
    \fi
    \unless\ifcsname#1\endcsname
      \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
    \fi
```

We have to reset these flags here because both the naming and cross-referencing macros use the parser.

```
    \if@nameauth@OldReset
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
    \else
      \global\@nameauth@FullNamefalse%
      \global\@nameauth@FirstNamefalse%
    \fi
  \fi
}
```

`\@nameauth@West` Print Western names and "non-native" Eastern names from `\@nameauth@name` and `\AKA`. We inherit internal macros from the parser and do nothing apart from the locked state.

```
\newcommand*\@nameauth@West[1]
{%
  \if@nameauth@BigLock\@nameauth@Lockfalse\fi
  \if@nameauth@Lock
    \unless\ifcsname#1\endcsname
      \@nameauth@FirstFormattrue%
    \fi
    \if@nameauth@InAKA
      \if@nameauth@AltAKA
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNametrue%
      \else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \else
      \unless\ifcsname#1\endcsname
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \fi
    \if@nameauth@FirstName
```

```
      \@nameauth@FullNamefalse%
    \fi
    \if@nameauth@FullName
      \if@nameauth@RevThis
        \@nameauth@Hook{\SNN\SpaceX\FNN}%
      \else
        \if@nameauth@RevThisComma
          \edef\RevSpace{,\SpaceX}%
          \@nameauth@Hook{\SNN\RevSpace\FNN}%
        \else
          \@nameauth@Hook{\FNN\SpaceX\SNN}%
        \fi
      \fi
    \else
      \if@nameauth@FirstName
        \@nameauth@Hook{\FNN}%
      \else
        \@nameauth@Hook{\rootb}%
      \fi
    \fi
    \unless\ifcsname#1\endcsname
      \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
    \fi
```

We have to reset these flags here because both the naming and cross-referencing macros use the parser.

```
    \if@nameauth@OldReset
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
    \else
      \global\@nameauth@FullNamefalse%
      \global\@nameauth@FirstNamefalse%
    \fi
  \fi
}
```

Format Hook Dispatcher

`\@nameauth@Hook` Flags help the dispatcher invoke the correct formatting hooks. The flags control which hook is called (first / subsequent use, name type). The first set of tests handles formatting within `\AKA`. The second set of tests handles regular name formatting.

```
\newcommand*\@nameauth@Hook[1]
{%
  \if@nameauth@BigLock\@nameauth@Lockfalse\fi
  \if@nameauth@Lock
    \@nameauth@InHooktrue%
    \protected@edef\test{#1}%
    \expandafter\@nameauth@TestDot\expandafter{\test}%
    \if@nameauth@InAKA
      \if@nameauth@AlwaysFormat
        \@nameauth@FirstFormattrue%
      \else
        \unless\if@nameauth@AKAFormat
        \@nameauth@FirstFormatfalse\fi
      \fi
      \if@nameauth@MainFormat
        \if@nameauth@FirstFormat
```

```
          \bgroup\NamesFormat{#1}\egroup%
        \else
          \bgroup\MainNameHook{#1}\egroup%
        \fi
      \else
        \if@nameauth@FirstFormat
          \bgroup\FrontNamesFormat{#1}\egroup%
        \else
          \bgroup\FrontNameHook{#1}\egroup%
        \fi
      \fi
    \else
      \if@nameauth@AlwaysFormat
        \@nameauth@FirstFormattrue%
      \fi
      \if@nameauth@MainFormat
        \if@nameauth@FirstFormat
          \bgroup\NamesFormat{#1}\egroup%
        \else
          \bgroup\MainNameHook{#1}\egroup%
        \fi
      \else
        \if@nameauth@FirstFormat
          \bgroup\FrontNamesFormat{#1}\egroup%
        \else
          \bgroup\FrontNameHook{#1}\egroup%
        \fi
      \fi
    \fi
```

We have to reset this flag here because both the naming and cross-referencing macros use the parser.

```
    \if@nameauth@OldReset
      \@nameauth@FirstFormatfalse%
    \else
      \global\@nameauth@FirstFormatfalse%
    \fi
    \@nameauth@InHookfalse%
  \fi
}
```

Indexing Internals

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

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

```
          \protected@edef\Entry{\Pre#2\Tag}%
        \fi
      \else
        \if@nameauth@Xref
          \protected@edef\Entry{#2\ShortTag}%
        \else
          \protected@edef\Entry{#2\Tag}%
        \fi
      \fi
    \else
      \ifcsname#1!PRE\endcsname
        \protected@edef\Pre{\csname#1!PRE\endcsname}%
        \protected@edef\Entry{\Pre#2}%
      \else
        \protected@edef\Entry{#2}%
      \fi
    \fi
    \ex\index\ex{\Entry}%
  \fi
}
```

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

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

Debugging Help

`\@nameauth@Debug` This Swiss-army knife for debugging shows name control sequence patterns, full index entries with tags, and short index entries with just the name. Other macros call it to get the desired info. We set up a local scope, redefine `\index` to print an argument in the text, force indexing to occur, and ignore whether we are working with xrefs.

```
\newcommandx*\@nameauth@Debug[3][1=\@empty, 3=\@empty]
{%
  \bgroup%
    \def\index##1{##1}%
    \@nameauth@DoIndextrue%
    \protected@edef\arga{\trim@spaces{#1}}%
    \protected@edef\argc{\trim@spaces{#3}}%
    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
    \def\csb{\@nameauth@Clean{#2}}%
    \def\csbc{\@nameauth@Clean{#2,#3}}%
    \def\csab{\@nameauth@Clean{#1!#2}}%
    \@nameauth@Error{#2}{macro \string\@nameauth@Debug}%
```

We interleave printing name patterns (`\ShowPattern`), printing full index entries as if they were page refs (`\ShowIdxPageref*`), and printing short index entries (`\ShowIdxPageref`). Since we are in a local scope we delete the tag and xref control sequences as needed. They will be restored when the scope ends. We do not care about xrefs because we just want to see what happens with the name. We can always go to the `idx` and `ind` files if needed.

```
    \ifx\arga\@empty
      \ifx\argc\@empty
        \ifdefined\ShortIdxEntry
          \csundef{\csb!PRE}%
          \csundef{\csb!TAG}%
          \csundef{\csb!PN}%
          \IndexName[#1]{#2}[#3]%
        \else
          \ifdefined\LongIdxEntry
```

```
            \csundef{\csb!PN}%
            \IndexName[#1]{#2}[#3]%
          \else
            \csb%
          \fi
        \fi
      \else
        \ifx\suffb\@empty
          \ifdefined\ShortIdxEntry
            \csundef{\csbc!PRE}%
            \csundef{\csbc!TAG}%
            \csundef{\csbc!PN}%
            \IndexName[#1]{#2}[#3]%
          \else
            \ifdefined\LongIdxEntry
              \csundef{\csbc!PN}%
              \IndexName[#1]{#2}[#3]%
            \else
              \csbc%
            \fi
          \fi
        \else
          \ifdefined\ShortIdxEntry
            \csundef{\csb!PRE}%
            \csundef{\csb!TAG}%
            \csundef{\csb!PN}%
            \IndexName[#1]{#2}[#3]%
          \else
            \ifdefined\LongIdxEntry
              \csundef{\csb!PN}%
              \IndexName[#1]{#2}[#3]%
            \else
              \csb%
            \fi
          \fi
        \fi
      \fi
    \else
      \ifdefined\ShortIdxEntry
        \csundef{\csab!PRE}%
        \csundef{\csab!TAG}%
        \csundef{\csab!PN}%
        \IndexName[#1]{#2}[#3]%
      \else
        \ifdefined\LongIdxEntry
          \csundef{\csab!PN}%
          \IndexName[#1]{#2}[#3]%
        \else
          \csab%
        \fi
      \fi
    \fi
    \global\undef{\LongIdxEntry}%
    \global\undef{\ShortIdxEntry}%
  \egroup%
}
```

3.5 User Interface Macros: Prefix Macros

Syntactic Formatting — Capitalization

`\CapThis` Tells the root capping macro to cap the first character of all name elements.

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

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

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

`\CapName` Capitalize entire required name. Overrides `\CapThis` for surnames.

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

`\AllCapsInactive` Turn off global surname capitalization.

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

`\AllCapsActive` Turn on global surname capitalization. Activates `\CapName` for every name.

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

Syntactic Formatting — Reversing

`\RevName` Reverse name order.

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

`\ReverseInactive` Turn off global name reversing.

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

`\ReverseActive` Turn on global name reversing. Activates `\RevName` for every name.

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

`\ForceFN` Force the printing of an Eastern forename or ancient affix in the text, but only when using the "short name" macro `\FName` and the `\S`⟨*macro*⟩.

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

Syntactic Formatting — Reversing with Commas

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

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

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

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

`\ReverseCommaActive` Turn on global "last-name-comma-first." Activates `\RevComma` for every name.

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

Alternate Formatting

`\AltFormatActive` Turn on alternate formatting, engage the formatting macros.

```
\newcommand*\AltFormatActive{%
  \global\@nameauth@AltFormattrue%
  \global\@nameauth@DoAlttrue%
}
```

\AltFormatActive* Turn on alternate formatting, disengage the formatting macros.

```
\WithSuffix{\newcommand*}\AltFormatActive*{%
  \global\@nameauth@AltFormattrue%
  \global\@nameauth@DoAltfalse%
}
```

\AltFormatInactive Turn off alternate formatting altogether.

```
\newcommand*\AltFormatInactive{%
  \global\@nameauth@AltFormatfalse%
  \global\@nameauth@DoAltfalse%
}
```

\AltOn Locally turn on alternate formatting.

```
\newcommand*\AltOn{%
  \if@nameauth@InHook
    \if@nameauth@AltFormat\@nameauth@DoAlttrue\fi
  \fi
}
```

\AltOff Locally turn off alternate formatting.

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

\AltCaps Alternate discretionary capping macro triggered by \CapThis.

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

\textSC Alternate formatting macro: small caps when active.

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

\textUC Alternate formatting macro: uppercase when active.

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

\textIT Alternate formatting macro: italic when active.

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

\textBF Alternate formatting macro: boldface when active.

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

Syntactic Formatting — Affixes

\ShowComma Put comma between name and suffix one time.

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

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

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

\DropAffix Suppress the affix in a long Western name.

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

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

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

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

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

Post-Processing — Main Versus Front Matter

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

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

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

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

Name Decisions — First / Subsequent Reference

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

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

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

```
\newcommand*\SubvertThis{\@nameauth@Subvertttrue}
```

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

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

Name Occurrence Tweaks

\LocalNames `\LocalNames` sets `@nameauth@LocalNames` true so `\ForgetName` and `\SubvertName` do not affect both main and front matter naming systems.

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

\GlobalNames `\GlobalNames` sets `@nameauth@LocalNames` false. This restores the default behavior of `\ForgetName` and `\SubvertName`.

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

Index Operations

\IndexInactive Turn off global indexing of names.

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

\SkipIndex Turn off the next instance of indexing in `\Name`, `\FName`, and starred forms.

```
\newcommand*\SkipIndex{\@nameauth@SkipIndextrue}
```

\JustIndex Makes the next call to `\Name`, `\FName`, and starred forms act like `\IndexName`. Overrides `\SkipIndex`.

```
\newcommand*\JustIndex{\@nameauth@JustIndextrue}
```

\IndexActive Turn on global indexing of names.

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

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

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

\SeeAlso Change the type of cross-reference from a *see* reference to a *see also* reference. Works once per xref, unless one uses \Include*, in which case, take care!

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

3.6 User Interface Macros: General

\ShowPattern This displays the pattern that the name arguments generate; maybe useful for debugging.

```
\newcommand*\ShowPattern{\@nameauth@Debug}
```

\ShowIdxPageref This displays the index entry that will be generated. This may be useful for debugging.

```
\newcommand*\ShowIdxPageref%
  {\def\LongIdxEntry{}\ShowPattern}
```

\ShowIdxPageref* This displays an index entry with no tag. This may be useful for debugging.

```
\WithSuffix{\newcommand*}\ShowIdxPageref*%
  {\def\ShortIdxEntry{}\ShowPattern}
```

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

```
\newcommand*\NameParser
{%
  \if@nameauth@InHook
    \let\SNN\rootb%
    \ifx\arga\@empty
```

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

```
      \ifx\argc\@empty
        \let\FNN\suffb%
      \else
        \let\FNN\argc%
      \fi
      \ifx\suffb\@empty
```

Mononym case

```
        \ifx\FNN\@empty
          \SNN%
        \else
```

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

```
          \if@nameauth@FullName%
            \if@nameauth@RevThis
              \FNN\Space\SNN%
            \else
              \SNN\Space\FNN%
            \fi
          \else
            \if@nameauth@FirstName
              \if@nameauth@EastFN
                \FNN%
              \else
                \SNN%
              \fi
```

```
            \else
              \SNN%
            \fi
          \fi
        \fi
      \else
```

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

```
        \if@nameauth@FullName
          \if@nameauth@RevThis
            \FNN\Space\SNN%
          \else
            \SNN\Space\FNN%
          \fi
        \else
          \if@nameauth@FirstName
            \if@nameauth@EastFN
              \FNN%
            \else
              \SNN%
            \fi
          \else
            \SNN%
          \fi
        \fi
      \fi
    \else
```

Western name with name reversing and suffixes

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

```
  \fi
}
```

Traditional Naming Interface

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

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

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

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

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

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

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

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

Index Operations

\IndexProtect We shut down all output from the naming and indexing macros to protect against problems in the index in case a macro in an index entry should expand into one of the naming macros.

```
\newcommand*\IndexProtect
{%
  \@nameauth@DoIndexfalse%
  \@nameauth@BigLocktrue%
}
```

\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.

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

Test for malformed input.

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

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.

```
  \ifx\arga\@empty
    \ifx\argc\@empty
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IndexName: XRef: #2 exists}%
        \fi
      \else
        \ifx\suffb\@empty
```

mononym or Eastern / ancient name, new syntax

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

Eastern or ancient name, older syntax

```
          \@nameauth@Index{\csbc}{\rootb\space\argc}%
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexName: XRef: #2 exists}%
          \fi
        \else
```

Eastern or ancient name, new syntax, alternate name ignored

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

Western name, without and with affix

```
      \ifx\suffb\@empty
        \@nameauth@Index{\csab}%
        {\rootb,\space\arga}%
      \else
        \@nameauth@Index{\csab}%
        {\rootb,\space\arga,\space\suffb}%
      \fi
    \fi
  \fi
}
```

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

```
\newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\rootb{\@nameauth@Root{#2}}%
```

```
\protected@edef\suffb{\@nameauth@Suffix{#2}}%
\protected@edef\argc{\trim@spaces{#3}}%
\protected@edef\target{#4}%
\def\csb{\@nameauth@Clean{#2}}%
\def\csbc{\@nameauth@Clean{#2,#3}}%
\def\csab{\@nameauth@Clean{#1!#2}}%
\let\ex\expandafter%
```

Test for malformed input.

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

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

```
\ifx\arga\@empty
  \ifx\argc\@empty
    \ifcsname\csb!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #2 exists}%
      \fi
    \else
      \ifx\suffb\@empty
```

mononym or Eastern / ancient name, new syntax

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

Eastern or ancient name, older syntax

```
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csbc}%
            {\rootb\space\argc|seealso{\target}}%
        \else
          \@nameauth@Index{\csbc}%
            {\rootb\space\argc|see{\target}}%
        \fi
```

```
          \csgdef{\csbc!PN}{}%
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexRef: XRef: #2 exists}%
          \fi
        \else
```

Eastern or ancient name, new syntax, alternate name ignored

```
          \if@nameauth@SeeAlso
            \@nameauth@Index{\csb}%
              {\rootb\space\suffb|seealso{\target}}%
          \else
            \@nameauth@Index{\csb}%
              {\rootb\space\suffb|see{\target}}%
          \fi
          \csgdef{\csb!PN}{}%
        \fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #1 #2 exists}%
      \fi
    \else
```

Western name, without and with affix

```
      \ifx\suffb\@empty
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csab}%
          {\rootb,\space\arga|seealso{\target}}%
        \else
          \@nameauth@Index{\csab}%
          {\rootb,\space\arga|see{\target}}%
        \fi
      \else
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csab}%
          {\rootb,\space\arga,\space\suffb|seealso{\target}}%
        \else
          \@nameauth@Index{\csab}%
          {\rootb,\space\arga,\space\suffb|see{\target}}%
        \fi
      \fi
      \csgdef{\csab!PN}{}%
    \fi
  \fi
  \@nameauth@Xreffalse%
```

This may not be necessary, but we do it for consistency.

```
  \if@nameauth@OldReset
    \@nameauth@SeeAlsofalse%
  \else
    \global\@nameauth@SeeAlsofalse%
```

```
  \fi
}
```

\ExcludeName This macro prevents a name from being indexed.

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

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

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

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

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

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

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

```
  \@nameauth@Error{#2}{macro \string\IncludeName}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \ifcsname\csb!PN\endcsname
```

```
        \edef\testex{\csname\csb!PN\endcsname}%
        \unless\ifx\testex\@empty\global\csundef{\csb!PN}%
        \else
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IncludeName: Xref: #2 exists}%
          \fi
        \fi
      \fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \edef\testex{\csname\csbc!PN\endcsname}%
          \unless\ifx\testex\@empty\global\csundef{\csbc!PN}%
          \else
            \if@nameauth@Verbose
              \PackageWarning{nameauth}%
              {macro \IncludeName: Xref: #2 #3 exists}%
            \fi
          \fi
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \edef\testex{\csname\csb!PN\endcsname}%
          \unless\ifx\testex\@empty\global\csundef{\csb!PN}%
          \else
            \if@nameauth@Verbose
              \PackageWarning{nameauth}%
              {macro \IncludeName: Xref: #2 exists}%
            \fi
          \fi
        \fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \edef\testex{\csname\csab!PN\endcsname}%
      \unless\ifx\testex\@empty\global\csundef{\csab!PN}%
      \else
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IncludeName: Xref: #1 #2 exists}%
        \fi
      \fi
    \fi
  \fi
}
```

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

```
\WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
```

```
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
  \@nameauth@Error{#2}{macro \string\IncludeName*}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \global\csundef{\csb!PN}%
    \else
      \ifx\suffb\@empty
        \global\csundef{\csbc!PN}%
      \else
        \global\csundef{\csb!PN}%
      \fi
    \fi
  \else
    \global\csundef{\csab!PN}%
  \fi
}
```

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

```
\newcommandx*\PretagName[4][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

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

```
        \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \PretagName: tagging xref: #1 #2}%
      \fi
    \fi
    \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
  \fi
}
```

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

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

We parse the arguments, defining the macros used by \@nameauth@Index.

```
  \@nameauth@Error{#2}{macro \string\TagName}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging xref: #2}%
        \fi
      \else
        \csgdef{\csb!TAG}{#4}%
      \fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \TagName: not tagging xref: #2 #3}%
          \fi
        \else
          \csgdef{\csbc!TAG}{#4}%
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \TagName: not tagging xref: #2}%
          \fi
        \else
          \csgdef{\csb!TAG}{#4}%
        \fi
      \fi
    \fi
```

```
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \TagName: not tagging xref: #1 #2}%
      \fi
    \else
      \csgdef{\csab!TAG}{#4}%
    \fi
  \fi
}
```

\UntagName This deletes an index tag.

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

We parse the arguments, undefining the index tag macros.

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

Name Info Data Set: "Text Tags"

\NameAddInfo This creates a macro that expands to information associated with a given name, similar to an index tag, but usable in the body text.

```
\newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
  \@nameauth@Error{#2}{macro \string\NameAddInfo}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \csgdef{\csb!DB}{#4}%
```

```
    \else
      \ifx\Suff\@empty
        \csgdef{\csbc!DB}{#4}%
      \else
        \csgdef{\csb!DB}{#4}%
      \fi
    \fi
  \else
    \csgdef{\csab!DB}{#4}%
  \fi
}
```

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

```
\newcommandx*\NameQueryInfo[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
  \unless\if@nameauth@BigLock
```

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

```
    \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
    \ifx\arga\@empty
      \ifx\argc\@empty
        \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
      \else
        \ifx\Suff\@empty
          \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
        \else
          \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
        \fi
      \fi
    \else
      \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
    \fi
  \fi
}
```

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

```
\newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
  \@nameauth@Error{#2}{macro \string\NameClearInfo}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \global\csundef{\csb!DB}%
    \else
```

```
      \ifx\Suff\@empty
        \global\csundef{\csbc!DB}%
      \else
        \global\csundef{\csb!DB}%
      \fi
    \fi
  \else
    \global\csundef{\csab!DB}%
  \fi
}
```

Name Decisions

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

```
\newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

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

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

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

Below we parse the name arguments and choose the path.

```
  \@nameauth@Error{#2}{macro \string\IfFrontName}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
    \else
      \ifx\suffb\@empty
```

```
        \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
      \else
        \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!NF\endcsname{#4}\else{#5}\fi
  \fi
}
```

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

```
\newcommandx\IfAKA[6][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
  \@nameauth@Error{#2}{macro \string\IfAKA}%
  \ifx\arga\@empty
    \ifx\argc\@empty
      \ifcsname\csb!PN\endcsname
        \edef\testex{\csname\csb!PN\endcsname}%
        \ifx\testex\@empty{#4}\else{#6}\fi
      \else{#5}\fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \edef\testex{\csname\csbc!PN\endcsname}%
          \ifx\testex\@empty{#4}\else{#6}\fi
        \else{#5}\fi
      \else
        \ifcsname\csb!PN\endcsname
          \edef\testex{\csname\csb!PN\endcsname}%
          \ifx\testex\@empty{#4}\else{#6}\fi
        \else{#5}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \edef\testex{\csname\csab!PN\endcsname}%
      \ifx\testex\@empty{#4}\else{#6}\fi
    \else{#5}\fi
  \fi
}
```

Changing Name Decisions

`\ForgetName` This undefines a control sequence to force a "first use."

```
\newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\argc{\trim@spaces{#3}}%
```

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

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

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

```
  \fi
}
```

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

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

We make copies of the arguments to test them.

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

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

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

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

Alternate Names

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

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

Prevent entering `\AKA` via itself or `\@nameauth@Name`. Prevent the index-only flag. Tell the formatting system that `\AKA` is running.

```
  \if@nameauth@BigLock\@nameauth@Locktrue\fi
  \unless\if@nameauth@Lock
    \@nameauth@Locktrue%
    \@nameauth@InAKAtrue%
    \if@nameauth@OldReset
      \@nameauth@JustIndexfalse%
    \else
      \global\@nameauth@JustIndexfalse%
    \fi
```

Test for malformed input.

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

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

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

Create an index cross-reference based on the arguments.

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

```
    \fi
  \fi
```

Reset all the "per name" Boolean values. The default is global.

```
  \if@nameauth@OldReset
    \@nameauth@SkipIndexfalse%
    \@nameauth@AltAKAfalse%
    \@nameauth@NBSPfalse%
    \@nameauth@NBSPXfalse%
    \@nameauth@DoCapsfalse%
    \@nameauth@Accentfalse%
    \@nameauth@AllThisfalse%
    \@nameauth@ShowCommafalse%
    \@nameauth@NoCommafalse%
    \@nameauth@RevThisfalse%
    \@nameauth@RevThisCommafalse%
    \@nameauth@ShortSNNfalse%
    \@nameauth@EastFNfalse%
  \else
    \global\@nameauth@SkipIndexfalse%
    \global\@nameauth@AltAKAfalse%
    \global\@nameauth@NBSPfalse%
    \global\@nameauth@NBSPXfalse%
    \global\@nameauth@DoCapsfalse%
    \global\@nameauth@Accentfalse%
    \global\@nameauth@AllThisfalse%
    \global\@nameauth@ShowCommafalse%
    \global\@nameauth@NoCommafalse%
    \global\@nameauth@RevThisfalse%
    \global\@nameauth@RevThisCommafalse%
    \global\@nameauth@ShortSNNfalse%
    \global\@nameauth@EastFNfalse%
  \fi
  \@nameauth@Lockfalse%
  \@nameauth@InAKAfalse%
```

Close the "locked" branch.

```
  \fi
```

Call the full stop detection.

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

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

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

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

```
\newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]
{%
  \@nameauth@JustIndexfalse%
  \if@nameauth@SkipIndex
    \NameauthName[#1]{#2}\space(\SkipIndex\AKA[#1]{#2}[#3]{#4}[#5])%
  \else
    \NameauthName[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
  \fi
}
```

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

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

Simplified Interface

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

```
\newenvironment{nameauth}{%
  \begingroup%
  \let\ex\expandafter%
  \csdef{<}##1&##2&##3&##4>{%
    \protected@edef\@arga{\trim@spaces{##1}}%
    \protected@edef\@testb{\trim@spaces{##2}}%
    \protected@edef\@testd{\trim@spaces{##4}}%
    \@nameauth@etoksb\expandafter{##2}%
    \@nameauth@etoksc\expandafter{##3}%
    \@nameauth@etoksd\expandafter{##4}%
```

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

```
    \ifx\@arga\@empty
      \PackageError{nameauth}%
      {environment nameauth: Control sequence missing}%
    \fi
    \@nameauth@Error{##3}{environment nameauth}%
    \ifcsname\@arga\endcsname
      \PackageWarning{nameauth}%
      {environment nameauth: Shorthand macro already exists}%
    \fi
```

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.

```
    \ifx\@testd\@empty
      \ifx\@testb\@empty
        \ex\csgdef\ex{\ex\@arga\ex}%
          \ex{\ex\NameauthName\ex{\the\@nameauth@etoksc}}%
        \ex\csgdef\ex{\ex L\ex\@arga\ex}%
          \ex{\ex\@nameauth@FullNametrue%
          \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
        \ex\csgdef\ex{\ex S\ex\@arga\ex}%
          \ex{\ex\@nameauth@FirstNametrue%
          \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
      \else
```

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.

```
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
          \ex\ex\ex{\ex\ex\ex\NameauthName%
          \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
          \ex{\the\@nameauth@etoksc}}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
          \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
          \ex\ex\ex\NameauthLName%
          \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
          \ex{\the\@nameauth@etoksc}}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
```

```
          \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\NameauthFName%
          \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
          \ex{\the\@nameauth@etoksc}}%
      \fi
    \else
```

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.

```
      \ifx\@testb\@empty
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
          \ex\ex\ex{\ex\ex\ex\NameauthName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
          \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
          \ex\ex\ex\NameauthLName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
          \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\NameauthFName%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
      \else
```

Here are Western names with alternates. We have three arguments to expand, so we have seven, three, and one respective use of `\expandafter`.

```
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@arga\ex\ex\ex\ex\ex\ex\ex}%
          \ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\NameauthName%
          \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the%
          \ex\ex\ex\@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex L\ex\ex\ex\ex\ex\ex\ex\@arga%
          \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@nameauth@FullNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthLName%
          \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
          \@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
        \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex S\ex\ex\ex\ex\ex\ex\ex\@arga%
          \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
          \ex\ex\ex\ex\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\ex\ex\ex\ex\NameauthFName%
          \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
          \@nameauth@etoksb\ex\ex\ex]%
          \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
          \ex[\the\@nameauth@etoksd]}%
      \fi
    \fi\ignorespaces%
  }\ignorespaces%
}{\endgroup\ignorespaces}
```

4 Change History

0.7
General: Initial release 1
0.75
\ForgetName: Current args 108
\IndexName: Current args 96
0.85
\@nameauth@Name: Hide commas 81
\AKA: Hide commas 111
\IndexName: Hide commas 96
0.9
\@nameauth@@Suffix: Added 78
\@nameauth@Suffix: Added 78
\AKA*: Added 112
\FName: Added 96
\SubvertName: Added 110
0.94
\@nameauth@Index: Added 88
\CapThis: Added 91
\ExcludeName: Added 100
\IndexActive: Added 93
\IndexInactive: Added 93
1.0
General: Works with microtype, memoir . . . 1
1.2
\TagName: Added 104
\UntagName: Added 105
1.26
\AKA: Fix affixes 111
\IndexName: Fix affixes 96
1.4
\ShowComma: Added 92
1.5
\@nameauth@@Suffix: Trim spaces 78
\@nameauth@Name: Reversing / caps 81
\AKA: Reversing, caps 111
\AllCapsActive: Added 91
\AllCapsInactive: Added 91
\CapName: Added 91
\RevComma: Added 91
\RevName: Added 91
\ReverseActive: Added 91
\ReverseCommaActive: Added 91
\ReverseCommaInactive: Added 91
\ReverseInactive: Added 91
1.6
nameauth: Environment added 113
1.9
\ForgetName: Global undef 108
\KeepAffix: Added 93
\TagName: Fix cs collisions 104
\UntagName: Global undef, no cs collisions 105
nameauth: Bugfix 113
2.0
\@nameauth@@Root: Trim spaces 78
\@nameauth@Actual: Added 89
\@nameauth@Index: New tagging 88
\@nameauth@Name: Trim spaces; fix tags . 81
\AKA: Trim spaces; fix tagging 111
\IndexActual: Added 94
\IndexName: Fix spaces, tagging 96
\PretagName: Added 103
\TagName: Redesign tagging 104
\UntagName: Redesign tagging 105
General: Use dtxgen; prevent bad args . . . 1
nameauth: Better arg handling 113
2.1
\@nameauth@Name: Fix Unicode 81
\AKA: Fix Unicode 111
\AccentCapThis: Added 91
2.11
nameauth: Bugfix 113
2.2
\NameauthFName: Added 76
\NameauthName: Added 76
2.3
\@nameauth@Name: Internal 81
\AKA: Fix starred mode 111
\ExcludeName: New xref test 100
\FName: Interface macro 96
\FName*: Interface macro 96
\ForgetName: Global or local 108
\GlobalNames: Added 93
\IfAKA: Added 108
\IfFrontName: Added 107
\IfMainName: Added 107
\LocalNames: Added 93
\Name: Interface macro 96
\Name*: Interface macro 96
\NameauthLName: Added 76
\PName: Work with hooks 112
\SubvertName: Global or local 110
2.4
\@nameauth@Hook: Current form 87
\@nameauth@Name: Set token regs 81
\FrontNameHook: Added 76
\GlobalNames: Ensure global 93
\IfAKA: Test for excluded 108
\LocalNames: Ensure global 93
\MainNameHook: Added 76
\NameAddInfo: Added 105
\NameClearInfo: Added 106
\NameQueryInfo: Added 106
2.41
\@nameauth@Name: Fix token regs 81
\AKA: Fix token regs 111

nameauth: No local \newtoks 113
2.5
\@nameauth@Hook: Improve hooks 87
\@nameauth@Name: Fix old syntax 81
\FrontNamesFormat: Added 76
General: No default format 1
2.6
\@nameauth@Name: Better indexing 81
\AKA: Fix index commas 111
\IndexName: Fix commas 96
\NoComma: Added 93
3.0
\@nameauth@@Root: Redesigned 78
\@nameauth@@Suffix: New test 78
\@nameauth@@TrimTag: Added 78
\@nameauth@Error: Added 80
\@nameauth@Hook: Fix punct. detection . 87
\@nameauth@Name: Redesigned 81
\@nameauth@NonWest: Added 85
\@nameauth@Parse: Added 82
\@nameauth@TrimTag: Added 78
\@nameauth@UTFtest: Added 79
\@nameauth@West: Added 86
\AKA: Redesigned 111
\DropAffix: Added 93
\ExcludeName: Redesigned 100
\ForceFN: Added 91
\IfAKA: Redesigned 108
\IncludeName: Added 101
\IncludeName*: Added 102
\IndexName: Redesigned 96
\IndexRef: Added 97
\NameParser: Added 94
\SeeAlso: Added 94
3.01
\@nameauth@Error: Fixed 80
3.02
\@nameauth@NonWest: Restrict \ForceFN 85
3.03
\NameParser: First name only for short forms 94
3.1
\@nameauth@C@p: Added 79
\@nameauth@C@pUTF: Added 80
\@nameauth@Cap: Redesigned 79
\@nameauth@Name: New workflow 81
\@nameauth@Parse: New workflow, caps . 82
\@nameauth@UTFtest: Override bypasses test 79
\AKA: Can skip index 111
\AltCaps: Added 92
\AltFormatActive: Added 91
\AltFormatActive*: Added 92
\AltFormatInactive: Added 92
\AltOff: Added 92
\AltOn: Added 92
\ForceName: Added 93
\ForgetThis: Added 93
\IncludeName*: Fixed 102
\IndexName: Better tests 96
\IndexRef: Better tests 97
\JustIndex: Added 93
\KeepName: Added 93
\NameParser: Fix old syntax; add NBSP . 94
\NameQueryInfo: Short macro 106
\PName: Can skip index 112
\SkipIndex: Added 93
\SubvertName: Fix old syntax 110
\SubvertThis: Added 93
\textBF: Added 92
\textIT: Added 92
\textSC: Added 92
\textUC: Added 92
3.2
\@nameauth@@GetSuff: Added 78
\@nameauth@@Root: Renamed 78
\@nameauth@@Suffix: Renamed 78
\@nameauth@C@p: Renamed, use \MakeUppercase 79
\@nameauth@C@pUTF: Use \MakeUppercase 80
\@nameauth@Cap: Non-UTF 79
\@nameauth@CapUTF: Added 79
\@nameauth@GetSuff: Added 78
\@nameauth@Parse: Fix alt. format, affixes, use \MakeUppercase 82
\@nameauth@TestToks: Added 78
\@nameauth@TrimTag: Renamed 78
\@nameauth@UTFtest: Non-suffix only . . . 79
\@nameauth@UTFtestS: Added 79
\AltCaps: Use \MakeUppercase 92
\NameParser: Fix alt. format, affixes . . . 94
\textUC: Use \MakeUppercase 92
3.3
\@nameauth@Debug: added 89
\@nameauth@Index: Tags support hyperref 88
\@nameauth@Name: global flag reset 81
\@nameauth@NonWest: global flag reset . . 85
\@nameauth@West: global flag reset 86
\AKA: global flag reset 111
\ExcludeName: More accurate warnings . 100
\IncludeName: Added warnings 101
\IndexProtect: added 96
\IndexRef: global flag reset 97
\NameQueryInfo: lock added 106
\ShowIdxPageref: added 94
\ShowIdxPageref*: added 94
\ShowPattern: added 94
General: Update manual 1

5 Index

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

Symbols

\@nameauth@@GetSuff . . . 104
\@nameauth@@Root 97
\@nameauth@@Suffix 101
\@nameauth@@TrimTag . . . 99
\@nameauth@Actual 579
\@nameauth@C@p 151
\@nameauth@C@pUTF 154
\@nameauth@Cap 150
\@nameauth@CapUTF 153
\@nameauth@CheckDot . . . 168
\@nameauth@Clean 94
\@nameauth@Debug 580
\@nameauth@Error 175
\@nameauth@EvalDot 170
\@nameauth@GetSuff 103
\@nameauth@Hook 494
\@nameauth@Index 547
\@nameauth@Name 188
\@nameauth@NonWest 382
\@nameauth@Parse 263
\@nameauth@Root 96
\@nameauth@Suffix 100
\@nameauth@TestDot 156
\@nameauth@TestToks . . . 105
\@nameauth@TrimTag 98
\@nameauth@UTFtest 113
\@nameauth@UTFtestS . . . 130
\@nameauth@West 440
\@nameauth@toksa 45, *58*
\@nameauth@toksb 45, *58*
\@nameauth@toksc 45, *58*

A

\AccentCapThis *28*, 658
Adams, John, pres. 61, 66
Æthelred II, king 9, 15, 26, 44
\AKA *54*, 1560
\AKA* *54*, 1627
à Kempis, Thomas . . . . . .
. . *see* Thomas à Kempis
\AllCapsActive *27*, 662
\AllCapsInactive . . . *27*, 661
\AltCaps *37*, 695
\AltFormatActive . . . *35*, 673
\AltFormatActive* . . *35*, 677
\AltFormatInactive . *35*, 681
\AltOff *37*, 690
\AltOn *37*, 685
Andreae, Ioannes . . . . . .
. *see* d'Andrea, Giovanni
Anthony, Susan B. 50
Arai Akino 27
Aristotle 7, 9, 12, 67
Arouet, François-Marie . . .
. . . . . *see* Voltaire
Atatürk . . *see* Kemal, Mustafa
Attila the Hun 9, 41

B

Babbage, Charles 36
Bernard of Clairvaux 56
Bess, Good Queen . . . . . .
. . . . . *see* Elizabeth I
Boëthius 16
Burns, Robert 64

C

\CapName *27*, 660
\CapThis *28*, 657
Carnap, Rudolph 33, 52
Carter, J.E., Jr., pres. . . 18, 40
Carter, Jimmy . . . . . .
. . . . *see* Carter, J.E., Jr.
Chaplin, Charlie 50, 51
Chiang Kai-shek‡, pres. . . 11, 26
Cicero, M.T. . 21, 22, 29, 34, 67
Clemens, Samuel L. . . . . .
. . . . . *see* Twain, Mark
Colfax, Schuyler, v.p. . . . 48
Confucius 21, 22, 26, 34
cummings, e.e. 28

D

Dagobert I‡, king 11, 67
d'Andrea, Giovanni 24
Davis, Sammy, Jr. 61, 66
Demetrius I Soter, king . . .
. . . . . 12, 29, 57
De Pamele, Jacques 24
de Smet, Pierre-Jean . . 62, 66
de Soto, Hernando . . . 8, 9, 28
Doctor angelicus . . . . . .
. . . *see* Thomas, Aquinas
Doctor mellifluus . . . . . .
see Bernard of Clairvaux
Dongen, Marc van 69, 81
Dracula . . . . . *see* Vlad III
\DropAffix *25*, 711
DuBois, W.E.B. . . . . .
. . . *see* Du Bois, W.E.B.
Du Bois, W.E.B. 23, 44
du Cange *see* du Fresne, Charles
du Fresne, Charles 55

E

Einstein, Albert 21, 22, 34, 50
Elizabeth I, queen . . . . . .
. . 7, 8, 21, 22, 32, 34, 55
environments:
nameauth *7*, 1638
\ExcludeName *41*, 995

F

\FName *22*, 818
\FName* *22*, 819
foo§ 39
\ForceFN *22*, 666
\ForceName *34*, 718
\ForgetName *50*, 1442
\ForgetThis *50*, 716
\FrontNameHook *33*, 54
\FrontNamesFormat . . . *33*, 53
FUKUYAMA Takeshi 36

G

Garbo, Greta 36, 43
Ghazāli 70
\GlobalNames *50*, 720
Goethe, J.W. von 8, 25
Gossett, Louis, Jr. 25
Grant, Ulysses S., pres. . 48, 49
Gregorio, Enrico 69
Gregory I, pope 46, 55
Gregory the Great . . . . . .
. . . . . *see* Gregory I

H

Hammerstein, Oskar, II . 25, 26
Harnack, Adolf 12, 68
Harun al-Rashid 61, 66
Hearn, Lafcadio 55
Henry VIII‡, king 11–13
Hope, Bob 52, 54
Hope, Leslie Townes . . . . .
. . . . . *see* Hope, Bob
Humperdinck, E. (composer) 46
Humperdinck, E. (singer) . 46

I

\if@nameauth@InAKA *58*
\if@nameauth@InName . . . *58*
\IfAKA *52*, 1407
\IfFrontName *52*, 1384

\IfMainName *51*, 1361
\IncludeName *41*, 1087
\IncludeName* *41*, 1146
\IndexActive *39*, 724
\IndexActual *44*, 725
\IndexInactive *39*, 721
\IndexName *39*, 826
\IndexProtect *39*, 821
\IndexRef *40*, 888
Iron Mike . . . *see* Tyson, Mike
Ishida Yoko‡ 27

J

Janos, James *see* Ventura, Jesse
Jefferson, Thomas, pres. 38
Jesus Christ 53
\JustIndex *40*, 723

K

Kanno, Yoko† 27
\KeepAffix *25*, 712
\KeepName *25*, 713
Kemal, Mustafa 59
Keynes, John Maynard . . . 34
King, Martin Luther, Jr. 14, 15
Koizumi Yakumo . . . . . .
. . . . *see* Hearn, Lafcadio

L

Lao-tzu 55, 56
Leo I, pope 46
Leo the Great . . . . . *see* Leo I
Lewis, Clive Staples . . 5, 9, 22
Li Er . . . . . *see* Lao-tzu
\LocalNames *50*, 719
Louis XIV, king . . . 25, 40, 55
Lovelace, Ada 36
Lueck, Uwe 69, 80
Luecking, Dan 69
Łukasiewicz, Jan 44
Luther, Martin 38

M

Maimonides . . . . . .
. . . 42, *see also* Rambam
\MainNameHook *33*, 52
Malebranche, Nicolas 33
Medici, Catherine de' . . . 38
Mencius 61, 66
Meng Ke . . . . . *see* Mencius
Miyazaki Hayao . . 7, 9, 21, 22
Molnár, Frenec† 6, 26
Moses ben-Maimon . . . . . .
. . . . . *see* Maimonides

N

\Name *21*, 815
\Name* *21*, 816
Name, Lost § . . . . . perdit(47)
\NameAddInfo *48*, 1290
nameauth (environment) *7*, 1638
\NameauthFName 57, *67*
\NameauthLName 56, *67*
\NameauthName 55, *67*
\NameClearInfo *48*, 1338
\NameParser *63*, 733
\NameQueryInfo *48*, 1313
\NamesActive *33*, 715
\NamesFormat *33*, 51
\NamesInactive *33*, 714
\NoComma *25*, 710
Noguchi, Hideyo† 6, 9, 12, 26, 27

O

Oberdiek, Heiko . . . 46, 69, 78

P

Pamelius, Jacobus . . . . . .
. *see* De Pamele, Jacques
Patton, George S., Jr. . . . .
. . . . . 5, 9, 12, 32
Paul . . . . . *see* Saul of Tarsus
\PName *55*, 1628
\PName* *55*, 1637
\PretagName *44*, 1169

R

Rambam . . . . . .
. 42, *see also* Maimonides
\RevComma *26*, 667
\ReverseActive *26*, 665
\ReverseCommaActive *26*, 671
\ReverseCommaInactive *26*, 669
\ReverseInactive . . . *26*, 664
\RevName *26*, 663
Rockefeller, Jay . *see* Rockefeller, John David, IV
Rockefeller, John David, III 36
Rockefeller, John David, IV 5, 9, 63
Rühmann, Heinrich Wilhelm . . . *see* Rühmann, Heinz
Rühmann, Heinz 43

S

Saul of Tarsus 53
Schlicht, Robert 69
Scipio Africanus, Publius Cornelius 4, 30, 31
\SeeAlso *41*, 727
Sergius Paulus, Lucius 53
Shakespeare, William . . . 64
\ShowComma *25*, 709
\ShowIdxPageref *13*, 729
\ShowIdxPageref* . . . *13*, 731
\ShowPattern *13*, 728
\SkipIndex *39*, 722
Snel van Royen, R. 42
Snel van Royen, W. 42
Snellius *see* Snel van Royen, R.; Snel van Royen, W.
Stephani, Philipp 69
Strietelmeier, John 16
\SubvertName *50*, 1501
\SubvertThis *50*, 717
Sullenberger, Chesley B., III 22
Sun King . . . . . *see* Louis XIV
Sun Yat-sen, pres. 25, 26

T

\TagName *45*, 1216
\textBF *36*, 707
\textIT *36*, 705
\textSC *36*, 701
\textUC *36*, 703
Thomas à Kempis 28
Thomas Aquinas 44
TOKUGAWA Ieyasu 36
Twain, Mark 56
Tyson, Mike 23

U

\UntagName *45*, 1267

V

Van Buren, Martin, pres. . .
. . . . . 8, 26, 49
Ventura, Jesse 52
Vlad II Dracul 57
Vlad III Dracula 57
Vlad Ţepeş . . . . . *see* Vlad III
Voltaire 56

W

Washington, George, pres. .
5, 8, 9, 40, 48, 49, 59, 67
White, E. B. 14, 16
William I 56
William the Conqueror . . .
. . . . . *see* William I

Y

Yamamoto Isoroku . . . . . .
. . . . . 7, 9, 12, 27, 32
Yohko 27
Yoshida Shigeru‡, PM 11

Z

Ziegler, Caspar 24