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

Charles P. Schaum[†]

Released 2020/02/26

Abstract

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

Contents

---

[*]This file describes version 3.4, last revised 2020/02/26.
[†]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).

<table>
<tr><td colspan="4">**"Non-native" Eastern Names, Western Index Entry**</td></tr>
<tr><td></td><td>Required
forename(s)</td><td>Required
surname(s),
no ⟨*Affix*⟩</td><td>Optional,
in text only</td></tr>
<tr><td>`\Name`
`\Name*`
`\FName`</td><td>`[`⟨***FNN***⟩`]`</td><td>`{`⟨***SNN***⟩`}`</td><td>`[`⟨***Alternate***⟩`]`</td></tr>
</table>

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).

<table>
<tr><td colspan="3">**"Native" Eastern Names in the Text, Eastern Index Entry**</td></tr>
<tr><td></td><td>Required surname
and forename</td><td>Optional,
in text only</td></tr>
<tr><td>`\Name`
`\Name*`
`\FName`</td><td>`{`⟨***SNN, Affix***⟩`}`</td><td>`[`⟨***Alternate***⟩`]`</td></tr>
</table>

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.

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

1.4 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)
```

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

<div style="text-align:right">

`\ShowPattern`
3.3

</div>

`\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`

<div style="text-align:right">

`\ShowIdxPageref`
`\ShowIdxPageref*`
3.3

</div>

`\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:

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

14

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.

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.

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: BoÃ«thius[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 Ã« correspond to \IeC{\"e}.

16

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.

<div align="center">

Affect the Syntax of Names

</div>

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	\KeepName
		\RevCommaActive	\ForceFN
		\RevCommaInactive	\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:

```
1  \begin{nameauth}
2    \< Lewis & Clive Staples & Lewis              &        >
3    \< CSL   & Clive Staples & Lewis              & C.S.   >
4    \< Ches  & Chesley B.    & Sullenberger, III &        >
5    \< Sully & Chesley B.    & Sullenberger, III & Sully  >
6    \< Miyaz &               & Miyazaki, Hayao   &        >
7    \< MSens &               & Miyazaki, Hayao   & Sensei >
8  \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:

```
1  \begin{nameauth}
2    \< Tyson & Mike & Tyson & >
3    \< Iron & Mike & Tyson & Iron Mike >
4  \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:

```
1  \begin{nameauth}
2    \< DuBois    & W.E.B. & Du~Bois & >
3    \< AltDuBois & W.E.B. & DuBois  & >
4  \end{nameauth}
5  \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).

```
1  \PretagName[Jacques]{De~Pamele}{Depamele, Jacques}
2  \Name[Jacques]{De~Pamele}[Jacques de~Joigny]
3  \IndexRef[Jacobus]{Pamelius}{De~Pamele, Jacques}
4  \Name[Jacobus]{Pamelius}
5
6  \PretagName[Giovanni]{d'Andrea}{Dandrea, Giovanni}
7  \Name[Giovanni]{d'Andrea}
8  \IndexRef[Ioannes]{Andreae}{d'Andrea, Giovanni}
9  \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.

<div style="border:1px solid">

Back to Section 1.2

</div>

---

[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`   In the text only, `\KeepName` turns all spaces **between** name elements ⟨*FNN*⟩,
3.1   ⟨*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`   Preceding the naming macros with `\DropAffix` will suppress an affix in a
3.0   Western name. `\DropAffix\Name*[Oskar]{Hammerstein, II}` produces "Oskar Hammerstein." This does not affect non-Western names.

`\ShowComma`   `\ShowComma` forces a comma between a Western name and its affix. It works
`\NoComma`   like the `comma` option on a per-name basis, and only in the body text. `\NoComma`
2.6   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

25

2.3.2 Listing Western names by Surname

\ReverseCommaActive — In addition to the options for reversed comma listing (Section 2.1), the macros
\ReverseCommaInactive — \ReverseCommaActive and \ReverseCommaInactive function the same way with
\RevComma — 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
3.0 — 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.

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 exam-
ple, 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 — In addition to the options for reversing (Section 2.1), \ReverseActive and
\ReverseInactive — \ReverseInactive reverse name order for blocks of text. These all override the
\RevName — 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 differ-
ences 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 In addition to the options for capitalizing (Section 2.1), `\AllCapsActive` and
\AllCapsInactive `\AllCapsInactive` work for blocks of text. All override `\CapName`, which works
\CapName 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 We recommend inserting a tilde (active character for a non-breaking space)
spaces 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*.

27

\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   If one uses this package on a system that does not handle Unicode, one can
3.0   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]

```
1  \NameAddInfo{Demetrius, I}{ Soter}
2  \PretagName{Demetrius, I}{Demetrius 1}
3  \TagName{Demetrius, I}{ Soter, king}
4  \begin{nameauth}
5    \< Dem & & Demetrius, I & >
6  \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).

```
7   \makeatletter
8   \renewcommand*\NamesFormat[1]{%
9     \begingroup%
10    \protected@edef\temp{\endgroup%
11      {#1\noexpand\NameQueryInfo
12        [\unexpanded\expandafter{\the\@nameauth@toksa}]
13        {\unexpanded\expandafter{\the\@nameauth@toksb}}
14        [\unexpanded\expandafter{\the\@nameauth@toksc}]%
15      }%
16    }%
17    \temp%
18  }
19  \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 show how to handle Roman names more properly.

---

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

Roman names have a *praenomen*, a personal name, then a *nomen*, a clan name, followed by a *cognomen*, a "nickname," except it could be inherited from one's father 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 to become one or two components: *praenomen* and *nomen*; *cognomen* and *agnomen*. We begin by defining a name with macros using \noexpand to prevent error:

```
1  \begin{nameauth}
2    \< Scipio & \noexpand\SCIPi & \noexpand\SCIPii & >
3  \end{nameauth}
4  \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:

```
5  \newif\ifSkipGens
6  \newif\ifNoGens
7  \newif\ifSkipAgnomen
8  \newif\ifNoAgnomen
9  \newcommand*\SCIPi{\ifNoGens Publius\else
10                    Publius Cornelius\fi}
11  \newcommand*\SCIPii{\ifNoAgnomen Scipio\else
12                     Scipio Africanus\fi}
13  \newcommand*\ScipioOnly{\SkipAgnomentrue\Scipio}
14  \renewcommand*\NamesFormat[1]%^^A
15    {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
16    \global\SkipGensfalse\global\SkipAgnomenfalse}
17  \renewcommand*\MainNameHook[1]%^^A
18    {\ifSkipGens\NoGenstrue\fi\ifSkipAgnomen\NoAgnomentrue\fi#1%
19    \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 and formatting hooks from above. We redefine the name in the following manner:

```
1  \begin{nameauth}
2    \< OScipio & Publius & \noexpand\CSA & >
3  \end{nameauth}
4  \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:

```
1  \newcommand*\CSA{\ifNoGens
2                    \ifNoAgnomen
3                      Scipio\else
4                      Scipio Africanus\fi
5                  \else\ifNoAgnomen
6                      Cornelius Scipio\else
7                      Cornelius Scipio Africanus\fi\fi}
8  \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.

<div style="border:1px solid; border-radius:8px; padding:8px; text-align:center;">

Back to Section 1.2

</div>

2.4 Basic Formatting

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

Full Forms, Front Matter		Short Forms, Front matter	
\NamesInactive		**\NamesInactive**	
First Use (Default)		**Later Use (Default)**	
\Name	George S. Patton Jr. Elizabeth I Yamamoto Isoroku	\Name	Patton; Elizabeth Yamamoto
\Name*	George S. Patton Jr.	\FName,	George S.; Elizabeth
\FName	George S. Patton Jr.	\S⟨*macro*⟩	Yamamoto
Later Use (* or \L⟨*macro*⟩)		**Later Use (\ForceName)**	
\Name*	George S. Patton Jr. Elizabeth I Yamamoto Isoroku	\Name	Patton; Elizabeth Yamamoto
		\FName,	George S.; Elizabeth
		\S⟨*macro*⟩	Yamamoto
Long, with \DropAffix		**Later Use (\ForceFN)**	
\DropAffix\LPat George S. Patton		\FName, \S⟨*macro*⟩	Isoroku

Full Forms, Main Matter		Short Forms, Main Matter	
\NamesActive		**\NamesActive**	
First Use (Default)		**Later Use (Default)**	
\Name	George S. Patton Jr. Elizabeth I Yamamoto Isoroku	\Name	Patton; Elizabeth Yamamoto
\Name*	George S. Patton Jr.	\FName,	George S.; Elizabeth
\FName	George S. Patton Jr.	\S⟨*macro*⟩	Yamamoto
Later Use (* or \L⟨*macro*⟩)		**Later Use (\ForceName)**	
\Name*	George S. Patton Jr. Elizabeth I Yamamoto Isoroku	\Name	Patton; Elizabeth Yamamoto
		\FName,	George S.; Elizabeth
		\S⟨*macro*⟩	Yamamoto
Long, with \DropAffix		**Later Use (\ForceFN)**	
\DropAffix\LPat George S. Patton		\FName, \S⟨*macro*⟩	Isoroku

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:

```
1  \colorlet{nared}{red!50!black}
2  \colorlet{nagreen}{green!35!black}
3  \colorlet{nablue}{blue!50!black}
4  \colorlet{nabrown}{brown!55!black}
5  \renewcommand*\FrontNamesFormat[1]{\color{nared}\sffamily #1}
6  \renewcommand*\FrontNameHook[1]{\color{nagreen}\sffamily #1}
7  \renewcommand*\NamesFormat[1]{\color{nablue}\sffamily #1}
8  \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:

```
1  \makeatletter
2   \let\@oldfntext\@makefntext
3   \long\def\@makefntext#1{\NamesInactive\@oldfntext{#1}\NamesActive}
4  \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:

```
5  \makeatletter
6   \let\@makefntext\@oldfntext
7  \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

34

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`.

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.

```
1  \PretagName[Greta]{\textSC{Garbo}}{Garbo, Greta}
2  \PretagName[Ada]{\textIT{Lovelace}}{Lovelace, Ada}
3  \PretagName[Charles]{\textBF{Babbage}}{Babbage, Charles}
4  \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.

A comma delimiter splits the mandatory macro argument into a root and an affix. To avoid errors, format the name and suffix separately.

```
1  \PretagName[John David]{\textSC{Rockefeller},\textSC{III}}
2    {Rockefeller, John David 3}
3  \PretagName{\textUC{Fukuyama}, Takeshi}{Fukuyama Takeshi}
4  \begin{nameauth}
5    \< JRIII & John David & \textSC{Rockefeller},\textSC{III} & >
6    \< Fukuyama  & & \textUC{Fukuyama}, Takeshi &          >
7    \< OFukuyama & & \textUC{Fukuyama}          & Takeshi >
8  \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{`$\langle Arg \rangle$`}`

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

```
1  \renewcommand*\MainNameHook%
2    {\color{nabrown}\sffamily\AltOff}% we match the manual
3
4  \IndexInactive
5  What's in a \Name{\noexpand\AltCaps{a} Name}?
6  \CapThis\Name{\noexpand\AltCaps{a} Name} smells not,
7  but a rose does. We avoid \Name{
8    \noexpand\textSC{\noexpand\AltCaps{a} Name},
9    \noexpand\textSC{Problem}}.
10 \CapThis\Name*{
11   \noexpand\textSC{\noexpand\AltCaps{a} Name},
12   \noexpand\textSC{Problem}} will not occur,
13    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:

```
1  \begin{nameauth}
2    \< Luth & Martin & \noexpand\textSC{Luther} & >
3  \end{nameauth}
4  \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":

```
1  \begin{nameauth}
2    \< Cath & Catherine \noexpand\AltCaps{d}e'
3         & \noexpand\textSC{Medici} & >
4  \end{nameauth}
5  \PretagName[Catherine \noexpand\AltCaps{d}e']
6              {\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]

```
1  \newif\ifGenitive
2  \newif\ifDoGen
3  \renewcommand*\NamesFormat[1]
4    {\ifGenitive\DoGentrue\fi#1\global\Genitivefalse}
5  \renewcommand*\MainNameHook[1]
6    {\ifGenitive\DoGentrue\fi\AltOff#1\global\Genitivefalse}
7  \begin{nameauth}
8    \< Jeff & Thomas & \noexpand\JEFF & >
9  \end{nameauth}
10 \PretagName[Thomas]{\noexpand\JEFF}{Jefferson, Thomas}
11 \TagName[Thomas]{\noexpand\JEFF}{, pres.|hyperpage}
12 \newcommand\JEFF{\ifDoGen\textSC{Jefferson's}\else
13   \textSC{Jefferson}\fi}
14
15 Consider \Genitivetrue\Jeff\ legacy. More on \Jeff\ later.
16 \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]

<div style="border:1px solid black; border-radius:10px; padding:10px; text-align:center;">

Back to Section 1.2

</div>

---

[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.

38

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 The noindex option deactivates the indexing function of this package until
\IndexInactive \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 Both the core name engine \@nameauth@Name and \AKA have locks that prevent
3.3 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 The prefix macro \SkipIndex will suppress indexing for just one instance of
3.1 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.

<div style="border:1px solid; border-radius:10px; text-align:center">

Back to Section 1.2

</div>

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.}`

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.

Advanced Cross-Referencing

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:

```
1  \AltFormatActive
2  \PretagName[Greta]{\textSC{Garbo}}{Garbo, Greta}
3  \PretagName[Heinz]{\textSC{Rühmann}}{Ruehmann, Heinz}
4  \PretagName[Heinrich Wilhelm]{\textSC{Rühmann}}%
5    {Ruehmann, Heinrich Wilhelm}
6
7  \newcommand{\RefArticle}[4]{%
8    \def\check{#2}%
9    \ifx\check\empty
10     \noindent\ForgetThis#1\ {#4}
11   \else
12     \noindent\ForceName#1\ ''\ForceName#2''
13     \ForceName#3\ {#4}
14   \fi
15 }
```

\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.

```
16 \RefArticle%
17   {\Name[Greta]{\textSC{Garbo}}}%
18   {}{}%
19   {(18 September 1905\,--\,15 April 1990) was a Swedish
20    film actress during the 1920s and 1930s.}
21
22 \RefArticle%
23   {%
24     \IndexRef[Heinrich Wilhelm]{\textSC{Rühmann}}%
25       {\textSC{Rühmann}, Heinz}%
26     \SubvertThis\FName[Heinrich Wilhelm]{\textSC{Rühmann}}%
27   }%
28   {\SubvertThis\FName[Heinz]{\textSC{Rühmann}}}%
29   {\Name[Heinz]{\textSC{Rühmann}}}%
30   {(7 March 1902\,--\,3 October 1994) was a German actor
31    in over 100 films.}
32
33 \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."

44

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⟩}`

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
```

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.

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.

3.3
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]

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:

```
1  \newcommand\Orphan[2]{#1(\hyperpage{#2})}
2  \TagName[Lost]{Name}{\,\S|Orphan{perdit}}
3  \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:

```
1  \newcommand*{\EndBreak}[1]{#1\newpage}
2  \makeatletter
3  \newcommand*{\MidBreak}[1]{#1\newpage\@gobble}
4  \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.

47

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]

```
 1  \documentclass{beamer}
 2  \usepackage{nameauth}
 3  \mode<presentation>
 4  \beamerdefaultoverlayspecification{<+->}
 5
 6  \begin{document}
 7
 8  \begin{frame}{Move Text Without Retyping Names}
 9    \begin{itemize}\footnotesize
10    \item<1-> Original\ForgetName[George]{Washington}%
11                    \ForgetName[George]{Washington's}\\
12            \Name[Martin]{Van Buren} changes
13            after the first overlay.
14    \begin{enumerate}
15    \item<2-> \IfMainName[George]{Washington's}{He}%
16            {\Name[George]{Washington}}
17            became the first president
18            of the United States.
19    \item<3-> \IfMainName[George]{Washington}{His}%
20            {\SkipIndex\Name*[George]{Washington's}}
21            military successes during the Seven Years War
22            readied him to command the army
23            of the Continental Congress.
24    \end{enumerate}
25    \item<1-> Reordered\ForgetName[George]{Washington}%
26                    \ForgetName[George]{Washington's}\\
27            \ForgetThis\Name[Ulysses S.]{Grant}
28            does not change.
29    \begin{enumerate}
30    \item<3-> \IfMainName[George]{Washington}{His}%
31            {\SkipIndex\Name*[George]{Washington's}}
32            military successes during the Seven Years War
33            readied him to command the army
34            of the Continental Congress.
35    \item<2-> \IfMainName[George]{Washington's}{He}%
36            {\Name[George]{Washington}}
37            became the first president
38            of the United States.
39    \end{enumerate}
40    \end{itemize}
41  \end{frame}
42
43  \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

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" state that will trigger a first-time name use:

> `\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, creating a name pattern control sequence and forcing a "subsequent use" case:

> `\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
```

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:

51

$$\text{\textbackslash IfMainName[Bob]\{Hope\}\{Bob here!\}\{No Bob.\}} \dots\dots\dots\dots \text{No Bob.}$$

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:

$$\text{\textbackslash IfMainName\{Elizabeth,I\}\{Bess here!\}\{No Bess.\}} \dots\dots\dots \text{Bess here!}$$

`\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:

```
1  \IfFrontName[Rudolph]{Carnap}%
2  {%
3    \IfMainName[Rudolph]{Carnap}%
4      {\Name[Rudolph]{Carnap} is both}%
5      {\Name[Rudolph]{Carnap} is only front-matter}%
6  }%
7  {%
8    \IfMainName[Rudolph]{Carnap}%
9      {\Name[Rudolph]{Carnap} is only main-matter}%
10     {\Name[Rudolph]{Carnap} is not mentioned}%
11 }
```

`\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:

```
1  \IfAKA[James]{Janos}%
2    {\Name*[Jesse]{Ventura} is a stage name}%
3    {\Name*[Jesse]{Ventura} is a regular name}%
4    {}
```

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

```
1  \IfAKA{Grinch}%
2    {\Name{Grinch} is an alias}%
3    {\Name{Grinch} is not an alias}%
4    {\Name{Grinch} is excluded}
```

We can combine all these macros create a complete test:

```
1  \IfAKA[FNN]{SNN, Affix}[Alternate]%
2    {true; it is a pseudonym}%
3    {% if not a pseudonym:
4      \IfFrontName[FNN]{SNN, Affix}[Alternate]% yes path
5      {\IfMainName[FNN]{SNN, Affix}[Alternate]%
6        {both}%
7        {front}%
8      }%
9      {\IfMainName[FNN]{SNN, Affix}[Alternate]% no path
10       {main}%
11       {does not exist}%
12     }%
13   }%
14   {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):

```
1  \IndexRef{Paul}{Saul of Tarsus}
2  \NameAddInfo{Saul, of Tarsus}{\IfMainName{Jesus, Christ}
3     {\IfMainName[Lucius]{Sergius Paulus}
4       {renames himself \Name{Paul}}
5       {a preacher to the Gentiles}}
6     {wrote that he persecuted Christians}}
7
8  \Name{Saul, of Tarsus} \NameQueryInfo{Saul, of Tarsus}. He
9  saw a vision of \Name{Jesus, Christ} on the road to Damascus
10 and became \NameQueryInfo{Saul, of Tarsus}. After converting
11 \Name[Lucius]{Sergius Paulus}, \Name{Saul, of Tarsus}
12 \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 (*also known as*) and its starred form display an alias in the text and
\AKA* 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 These convenience macros (an early feature) print a main name and a cross-
\PName* reference in parentheses:

> `\PName [⟨FNN⟩]{⟨SAFX⟩}[⟨xrefFNN⟩]{⟨xrefSAFX⟩}[⟨xrefAlternate⟩]`
> `\PName*[⟨FNN⟩]{⟨SAFX⟩}[⟨xrefFNN⟩]{⟨xrefSAFX⟩}[⟨xrefAlternate⟩]`

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 $\langle FNN \rangle \langle SAFX \rangle$, 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

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.

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

Within the document after the preamble:

```
8  Name{Vlad, III}[III Dracula], known as
9  \AKA{Vlad III}{Vlad, Ţepeş} (the Impaler)
10 after his death, was the son of \Name{Vlad, II}[II Dracul],
11 a member of the Order of the Dragon. Later references to
12 ''\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.

```
13 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.

<div style="border:1px solid #000;border-radius:8px;text-align:center;padding:8px">Back to Section 1.2</div>

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`:

```
 1  \newif\ifNoTag% allows us to work around \ForgetName
 2  \let\OldFormat\NamesFormat
 3  \let\OldFrontFormat\FrontNamesFormat
 4  \makeatletter
 5  \renewcommand*\NamesFormat[1]{\begingroup%
 6    \protected@edef\temp{\endgroup\textsc{#1}%
 7    \unless\ifNoTag
 8      \if@nameauth@InName
 9        {\bfseries\noexpand\NameQueryInfo
10        [\unexpanded\expandafter{\the\@nameauth@toksa}]
11        {\unexpanded\expandafter{\the\@nameauth@toksb}}
12        [\unexpanded\expandafter{\the\@nameauth@toksc}]}\fi
13      \if@nameauth@InAKA
14        {\normalfont\noexpand\NameQueryInfo
15        [\unexpanded\expandafter{\the\@nameauth@toksa}]
16        {\unexpanded\expandafter{\the\@nameauth@toksb}}
17        [\unexpanded\expandafter{\the\@nameauth@toksc}]}\fi
18    \fi}\temp\global\NoTagfalse%
19  }
20  \makeatother
21  \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:

```
22  \NameAddInfo[George]{Washington}{ (1732--99)}
23  \NameAddInfo[Mustafa]{Kemal}{ (1881--1938)}
24  \NameAddInfo{Atatürk}{ (in 1934, a special surname)}
25  \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:

```
26  \ForgetThis\Wash held office 1789--97.
27  No tags: \Wash.\\
28  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:

```
29  \Name[Mustafa]{Kemal} was granted the name
30  \Name{Atatürk}. We mention \Name[Mustafa]{Kemal}
31  and \Name{Atatürk} again.
32
33  First use, no tag:
34  \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`:

```
35  \makeatletter\@nameauth@AKAFormattrue\makeatother
36  \ForgetThis\Name[Mustafa]{Kemal} was granted the name
37  \ForceName\AKA[Mustafa]{Kemal}{Atatürk}. We mention
38  \Name[Mustafa]{Kemal} and \AKA[Mustafa]{Kemal}{Atatürk} again.
39
40  First use, no tag:
41  \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:

```
42  \let\NamesFormat\OldFormat
43  \let\FrontNamesFormat\OldFrontFormat
```

Back to Section

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`:

```
1  \newcommand*\textSC[1]{%
2    \if@nameauth@DoAlt\textsc{#1}\else#1\fi
3  }
```

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:

```
4  \newcommand*\AltOff{%
5    \if@nameauth@InHook\@nameauth@DoAltfalse\fi
6  }
```

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

```
7  \newcommand*\AltCaps[1]{%
8    \if@nameauth@InHook
9      \if@nameauth@DoCaps\MakeUppercase{#1}\else#1\fi
10   \else#1\fi
11 }
```

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:

```
12  \renewcommand*\MainNameHook{\AltOff}
13  \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:

```
14  \begin{nameauth}
15    \< Adams   & John  & \noexpand\textSC{Adams}      & >
16    \< SDJR    & Sammy & \noexpand\textSC{Davis},
17                        \noexpand\textSC{Jr}.          & >
18    \< HAR     &       & Harun, \noexpand\textSC%
19                        {\noexpand\AltCaps{a}l-Rashid} & >
20    \< Mencius &       & \noexpand\textSC{Mencius}     & >
21  \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:

```
22  \PretagName[John]{\noexpand\textSC{Adams}}{Adams, John}
23  \PretagName[Sammy]%
24    {\noexpand\textSC{Davis}, \noexpand\textSC{Jr}.}%
25    {Davis, Sammy, Jr.}
26  \PretagName{Harun, \noexpand\textSC%
27    {\noexpand\AltCaps{a}l-Rashid}}{Harun al-Rashid}
28  \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.

```
29  \PretagName{\noexpand\textSC{Meng}, Ke}{Meng Ke}
30  \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:

```
1  \makeatletter
2  \newcommand*\Fbox[1]{%
3    \if@nameauth@DoAlt\protect\fbox{#1}\else#1\fi
4  }
5  \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:

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

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

```
8  \PretagName[Pierre-Jean]%
9    {\noexpand\Fbox{\noexpand\AltCaps{d}e Smet}}%
10   {de Smet, Pierre-Jean}
11
12 \begin{nameauth}
13   \< deSmet & Pierre-Jean &
14     \noexpand\Fbox{\noexpand\AltCaps{d}e Smet} & >
15 \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:

```
1  \makeatletter
2  \renewcommand*\NamesFormat[1]{\small%
3    \hbox to 3.5em{[now]\hfill}\space\NameParser\\%
4    \@nameauth@FullNametrue%
5    \hbox to 3.5em{[long]\hfill}\space\NameParser\\%
6    \@nameauth@FullNamefalse%
7    \@nameauth@FirstNametrue%
8    \hbox to 3.5em{[short]\hfill}\space\NameParser}
9  \makeatother
10 \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:

```
1  \begin{nameauth}
2    \< Shak & \noexpand\WM & \noexpand\SHK & >
3  \end{nameauth}
4  \PretagName[\noexpand\WM]{\noexpand\SHK}{Shakespeare, William}
5  \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:

```
6  \newif\ifSpecialFN
7  \newif\ifSpecialSN
8  \newif\ifRevertSN
9  \newcommand*\WM{\ifSpecialFN Wm.\else William\fi}
10 \newcommand*\SHK{\ifRevertSN \textSC{Shakespeare}\else
11               \ifSpecialSN \noexpand\AltCaps{t}he Bard\else
12               \textSC{Shakespeare}\fi\fi}
13 \newcommand*\Revert{\RevertSNtrue}
14 \makeatletter
```

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

```
15 \renewcommand*\NamesFormat[1]{%
16   \RevertSNfalse\SpecialFNfalse\SpecialSNfalse#1%
17   \unless\ifinner\marginpar{%
18     \footnotesize\raggedleft%
19     \@nameauth@FullNametrue%
20     \@nameauth@FirstNamefalse%
21     \@nameauth@EastFNfalse%
22     \SpecialFNtrue\SpecialSNfalse%
23     \NameParser}%
24   \fi\global\RevertSNfalse}
25 \renewcommand*\MainNameHook[1]{%
26   \AltOff\SpecialFNfalse\SpecialSNtrue#1%
27   \unless\ifinner
28     \unless\ifRevertSN
29       \marginpar{%
30       \footnotesize\raggedleft%
31       \@nameauth@FullNamefalse%
32       \@nameauth@FirstNamefalse%
33       \@nameauth@EastFNfalse%
34       \SpecialFNfalse\SpecialSNfalse%
35       \NameParser}%
36     \fi
37   \fi\global\RevertSNfalse}
38 \makeatother
```

Wm. SHAKESPEARE   William SHAKESPEARE \Shak is the national poet of England in much
Robert BURNS   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
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.

64

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`:

```
1  \newif\ifFbox% Replaces \@nameauth@DoAlt
2  \newif\ifFirstCap% Replaces \@nameauth@DoCaps
3  \newif\ifInHook% Replaces \@nameauth@InHook
4  \Fboxtrue
```

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

```
5  \renewcommand*\Fbox[1]{%
6    \ifFbox\protect\fbox{#1}\else#1\fi
7  }
```

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

```
8   \renewcommand*\AltCaps[1]{%
9     \ifInHook
10      \ifFirstCap\MakeUppercase{#1}\else#1\fi
11    \else
12      #1%
13    \fi
14  }
```

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

```
15  \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:

```
16  \renewcommand*\NamesFormat[1]
17    {\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`:

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

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

```
20  \let\FrontNamesFormat\Namesformat
21  \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:

```
1   \newif\ifCaps
2   \Capstrue
3   \renewcommand*\textSC[1]{%
4     \ifCaps\textsc{#1}\else#1\fi
5   }
6   \renewcommand*\MainNameHook[1]
7   {%
8     \Capsfalse\InHooktrue\NameParser%
9     \global\FirstCapfalse%
10  }
11  \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

66

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:

```
1  \makeatletter
2  \newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
3    \protected@edef\a{\trim@spaces{#1}}%
4    \protected@edef\b{\trim@spaces{#2}}%
5    \protected@edef\c{\trim@spaces{#3}}%
6    \ifx\b\empty fail \else
7      \ifx\a\empty
8        \ifx\c\empty \hbox to 5em{Mononym:\hfill} {\b}\else
9        \hbox to 5em{Eastern:\hfill} {\b\ \c}\fi
10     \else
11       \ifx\c\empty \hbox to 5em{Western:\hfill} {\a\ \b}\else
12       \hbox to 5em{Alternate:\hfill} {\c\ \b}\fi
13     \fi
14   \fi
15   \global\@nameauth@FullNamefalse%
16   \global\@nameauth@FirstNamefalse%
17 }
18 \makeatother
19 \let\MyLName\MyName
20 \let\MyFName\MyName
21 \renewcommand*\NameauthName{\MyName}
22 \renewcommand*\NameauthLName{\MyLName}
23 \renewcommand*\NameauthFName{\MyFName}
24 \IndexName[George]{Washington}
25 \IndexName[M.T.]{Cicero}
26 \IndexName{Dagobert}[I]
27 \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).

67

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

```
 1  \makeatletter
 2  \newcommandx*\MyName[3][1=\@empty, 3=\@empty]{%
 3    \@nameauth@toksa\expandafter{#1}%
 4    \@nameauth@toksb\expandafter{#2}%
 5    \@nameauth@toksc\expandafter{#3}%
 6    \hbox to 4em{Normal: \hfill}%
 7    \fcolorbox{black}{gray!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
 8  }
 9  \newcommandx*\MyLName[3][1=\@empty, 3=\@empty]{%
10    \@nameauth@toksa\expandafter{#1}%
11    \@nameauth@toksb\expandafter{#2}%
12    \@nameauth@toksc\expandafter{#3}%
13    \hbox to 4em{Long: \hfill}%
14    \fcolorbox{black}{green!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
15  }
16  \newcommandx*\MyFName[3][1=\@empty, 3=\@empty]{%
17    \@nameauth@toksa\expandafter{#1}%
18    \@nameauth@toksb\expandafter{#2}%
19    \@nameauth@toksc\expandafter{#3}%
20    \hbox to 4em{Short: \hfill}%
21    \fcolorbox{black}{yellow!25!white}{\@nameauth@Name[#1]{#2}[#3]}%
22  }
23  \makeatother
24  \renewcommand*\NamesFormat[1]
25    {\hbox to 9em{\hfil\scshape#1\hfil}}
26  \renewcommand*\MainNameHook[1]{\hbox to 9em{\hfil#1\hfil}}
27  \renewcommand*\NameauthName{\MyName}
28  \renewcommand*\NameauthLName{\MyLName}
29  \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

68

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 Æ Ȳ	ā · ē ḡ · ī ō ū æ 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 ǰ 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 Ş Ţ Þ	ø œ ő ů ű ş ţ 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 Congrefs, July 4, 1776''`:

```
1  \usepackage[utf8]{inputenc} % For older TL releases
2  \usepackage[TS1,T1]{fontenc}
3  \usepackage{lmodern}% Contains TS1 glyph 115
4  \usepackage{newunicodechar}
5  \DeclareTextSymbolDefault{\textlongs}{TS1}
6  \DeclareTextSymbol{\textlongs}{TS1}{115}
7  \newunicodechar{f}{\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 Ghazālī 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.

```
1  \ifPDFTeX
2    \IfFileExists{utf8-2018.def}%
3      {\Name{Ghazāli}}{\Name{Ghazali}}%
4    \else\Name{Ghazāli}%
5  \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]

```
1  \IfFileExists{iftex.sty}{\usepackage{iftex}}{}
2  \unless\ifdefined\RequireTUTeX
3    \usepackage{ifxetex}
4    \usepackage{ifluatex}
5    \usepackage{ifpdf}
6  \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.

```
7  \newif\ifDoTikZ                        % Perhaps not needed
8  \ifxetex
9    \usepackage{fontspec}
10   \usepackage{polyglossia}
11   \setdefaultlanguage{american}        % Use own language
12   \usepackage{tikz}
13   \DoTikZtrue                          % Perhaps not needed
14 \else
15   \ifluatex
16     \ifpdf
17       \usepackage{fontspec}
18       \usepackage{polyglossia}
19       \setdefaultlanguage{american}    % Use own language
20       \usepackage{tikz}
21       \DoTikZtrue                      % Perhaps not needed
22     \else
23       \IfFileExists{utf8-2018.def}{}
24       {\usepackage[utf8]{inputenc}}
25       \usepackage[TS1,T1]{fontenc}
26       \usepackage[american]{babel}     % Use own language
27       \usepackage{lmodern}
28       % Perhaps add \usepackage{tikz}
29     \fi
30   \else
31     \IfFileExists{utf8-2018.def}{}
32     {\usepackage[utf8]{inputenc}}
33     \usepackage[TS1,T1]{fontenc}
34     \usepackage[american]{babel}       % Use own language
35     \usepackage{lmodern}
36     \ifpdf                             % Perhaps not needed
37       \usepackage{tikz}
38       \DoTikZtrue                      % Perhaps not needed
39     \fi
40   \fi
41 \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:

```
 1  \ifxetex xelatex%
 2  \else
 3    \ifluatex
 4      \ifpdf lualatex (pdf)%
 5      \else lualatex (dvi)%
 6      \fi
 7    \else
 8      \ifpdf pdflatex%
 9      \else latex (dvi)%
10      \fi
11    \fi
12  \fi
```

```
 1  \unless\ifxetex
 2    \unless\ifluatex
 3      \ifpdf pdflatex%
 4      \else latex (dvi)%
 5      \fi
 6    \else
 7      \ifpdf lualatex (pdf)%
 8      \else lualatex (dvi)%
 9      \fi
10    \fi
11  \else xelatex%
12  \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.

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

```
4 \newif\if@nameauth@Lock
5 \newif\if@nameauth@BigLock
6 \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`:

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

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

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

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

```
11 \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.

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

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

```
14 \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`.

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

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

```
17 \newif\if@nameauth@Forget
18 \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.

```
19 \newif\if@nameauth@FirstFormat
20 \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.

```
21 \newif\if@nameauth@AlwaysComma
22 \newif\if@nameauth@ShowComma
23 \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.

```
24 \newif\if@nameauth@NBSP
25 \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.

```
26 \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.

```
27 \newif\if@nameauth@FullName
28 \newif\if@nameauth@FirstName
29 \newif\if@nameauth@AltAKA
30 \newif\if@nameauth@OldAKA
31 \newif\if@nameauth@ShortSNN
32 \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.

```
33 \newif\if@nameauth@RevAll
34 \newif\if@nameauth@RevThis
35 \newif\if@nameauth@AllCaps
36 \newif\if@nameauth@AllThis
```

Last-Comma-First

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

```
37 \newif\if@nameauth@RevAllComma
38 \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.

```
39 \newif\if@nameauth@DoCaps
40 \newif\if@nameauth@UTF
41 \newif\if@nameauth@Accent
42 \newif\if@nameauth@AltFormat
43 \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.

```
44 \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.

```
45 \newtoks\@nameauth@toksa%
46 \newtoks\@nameauth@toksb%
47 \newtoks\@nameauth@toksc%
```

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

```
48 \newtoks\@nameauth@etoksb%
49 \newtoks\@nameauth@etoksc%
50 \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.

```
51 \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.

```
52 \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.

```
53 \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.

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

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

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

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

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

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

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

3.3 Package Options

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

```
58 \DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
59 \DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
60 \DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
61 \DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
62 \DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
63 \DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
64 \DeclareOption{oldreset}{\@nameauth@OldResettrue}
65 \DeclareOption{oldpass}{\@nameauth@OldPasstrue}
66 \DeclareOption{index}{\@nameauth@DoIndextrue}
67 \DeclareOption{noindex}{\@nameauth@DoIndexfalse}
68 \DeclareOption{pretag}{\@nameauth@Pretagtrue}
69 \DeclareOption{nopretag}{\@nameauth@Pretagfalse}
70 \DeclareOption{allcaps}{\@nameauth@AllCapstrue}
71 \DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
72 \DeclareOption{allreversed}%
73   {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
74 \DeclareOption{allrevcomma}%
75   {\@nameauth@RevAllfalse\@nameauth@RevAllCommatrue}
76 \DeclareOption{notreversed}%
77   {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
78 \DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
79 \DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
80 \DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
81 \DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
82 \DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
83 \DeclareOption{verbose}{\@nameauth@Verbosetrue}
84 \DeclareOption{altformat}{%
85   \@nameauth@AltFormattrue\@nameauth@DoAlttrue}
86 \ExecuteOptions%
87   {nocomma,mainmatter,index,pretag,%
88    normalcaps,notreversed,noformat}
89 \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.

```
90 \RequirePackage{etoolbox}
91 \RequirePackage{suffix}
92 \RequirePackage{trimspaces}
93 \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.

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

Parsing: Root and Suffix

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

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

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

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

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

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

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

```
99 \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.

```
100 \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.

```
101 \def\@nameauth@@Suffix#1,#2,#3\\%
102    {\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.

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

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

```
104 \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.

```
105 \newcommand*\@nameauth@TestToks[1]
106 {%
107    \toks@\expandafter{\@car#1\@nil}%
108    \edef\one{\the\toks@}%
109    \toks@\expandafter{\@carß\@nil}%
110    \edef\two{\the\toks@}%
111    \ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi
112 }
```

\@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.

```
113 \newcommand*\@nameauth@UTFtest[1]
114 {%
115   \def\testarg{#1}%
116   \ifdefined\Umathchar
117     \@nameauth@UTFfalse%
118   \else
119     \ifdefined\UTFviii@two@octets
120       \if@nameauth@Accent
121         \@nameauth@UTFtrue\@nameauth@Accentfalse%
122       \else
123         \expandafter\@nameauth@TestToks\expandafter{\testarg}%
124       \fi
125     \else
126       \@nameauth@UTFfalse%
127     \fi
128   \fi
129 }
```

\@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.

```
130 \newcommand*\@nameauth@UTFtestS[1]
131 {%
132   \let\ex\expandafter%
133   \ex\def\ex\testarg\ex{\@nameauth@GetSuff{#1}}%
134   \ex\toks@\ex\ex\ex{\testarg}%
135   \ex\def\ex\test@rg\ex{\the\toks@}%
136   \ifdefined\Umathchar
137     \@nameauth@UTFfalse%
138   \else
139     \ifdefined\UTFviii@two@octets
140       \if@nameauth@Accent
141         \@nameauth@UTFtrue\@nameauth@Accentfalse%
142       \else
143         \expandafter\@nameauth@TestToks\expandafter{\test@rg}%
144       \fi
145     \else
146       \@nameauth@UTFfalse%
147     \fi
148   \fi
149 }
```

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

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

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

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

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

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

79

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

```
154 \def\@nameauth@C@pUTF#1#2#3\\%
155   {\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.

```
156 \newcommand*\@nameauth@TestDot[1]
157 {%
158   \def\TestDot##1.\TestEnd##2\\{\TestPunct{##2}}%
159   \def\TestPunct##1{%
160     \ifx\TestPunct##1\TestPunct%
161     \else
162       \@nameauth@Puncttrue%
163     \fi
164   }%
165   \@nameauth@Punctfalse%
166   \TestDot#1\TestEnd.\TestEnd\\%
167 }
```

\@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.

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

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

```
170 \newcommand*\@nameauth@EvalDot%
171 {%
172   \let\@period=.%
173   \ifx\@token\@period\expandafter\@gobble \fi
174 }
```

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.

```
175 \newcommand*\@nameauth@Error[2]
176 {%
177   \edef\msga{#2 SNN arg empty}%
178   \edef\msgb{#2 SNN arg malformed}%
179   \protected@edef\testname{\trim@spaces{#1}}%
180   \protected@edef\testroot{\@nameauth@Root{#1}}%
181   \ifx\testname\@empty
182     \PackageError{nameauth}{\msga}%
183   \fi
184   \ifx\testroot\@empty
185       \PackageError{nameauth}{\msgb}%
186   \fi
187 }
```

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.

```
188 \newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
189 {%
```

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`.

```
190     \if@nameauth@BigLock\@nameauth@Locktrue\fi
191     \unless\if@nameauth@Lock
192       \@nameauth@Locktrue%
193       \@nameauth@InNametrue%
```

Test for malformed input.

```
194       \@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.

```
195       \if@nameauth@JustIndex
196         \IndexName[#1]{#2}[#3]%
197         \if@nameauth@OldPass
198           \@nameauth@JustIndexfalse%
199         \else
200           \if@nameauth@OldReset
201             \@nameauth@FullNamefalse%
202             \@nameauth@FirstNamefalse%
203             \@nameauth@JustIndexfalse%
204           \else
205             \global\@nameauth@FullNamefalse%
206             \global\@nameauth@FirstNamefalse%
207             \global\@nameauth@JustIndexfalse%
208           \fi
209         \fi
210       \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.

```
211       \if@nameauth@Forget
212         \ForgetName[#1]{#2}[#3]%
213       \else
214         \if@nameauth@Subvert
215           \SubvertName[#1]{#2}[#3]%
216         \fi
217       \fi
218       \leavevmode\hbox{}%
219       \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
220       \if@nameauth@MainFormat
221         \@nameauth@Parse[#1]{#2}[#3]{!MN}%
222       \else
223         \@nameauth@Parse[#1]{#2}[#3]{!NF}%
224       \fi
225       \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
```

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

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

```
228        \@nameauth@Forgetfalse%
229        \@nameauth@Subvertfalse%
230        \@nameauth@NBSPfalse%
231        \@nameauth@NBSPXfalse%
232        \@nameauth@DoCapsfalse%
233        \@nameauth@Accentfalse%
234        \@nameauth@AllThisfalse%
235        \@nameauth@ShowCommafalse%
236        \@nameauth@NoCommafalse%
237        \@nameauth@RevThisfalse%
238        \@nameauth@RevThisCommafalse%
239        \@nameauth@ShortSNNfalse%
240        \@nameauth@EastFNfalse%
241      \else
242        \global\@nameauth@SkipIndexfalse%
243        \global\@nameauth@Forgetfalse%
244        \global\@nameauth@Subvertfalse%
245        \global\@nameauth@NBSPfalse%
246        \global\@nameauth@NBSPXfalse%
247        \global\@nameauth@DoCapsfalse%
248        \global\@nameauth@Accentfalse%
249        \global\@nameauth@AllThisfalse%
250        \global\@nameauth@ShowCommafalse%
251        \global\@nameauth@NoCommafalse%
252        \global\@nameauth@RevThisfalse%
253        \global\@nameauth@RevThisCommafalse%
254        \global\@nameauth@ShortSNNfalse%
255        \global\@nameauth@EastFNfalse%
256      \fi
257    \fi
258    \@nameauth@Lockfalse%
259    \@nameauth@InNamefalse%
```

Close the "locked" branch.

```
260    \fi
```

Call the full stop detection.

```
261    \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
262 }
```

\@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.

```
263 \newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
264 {%
265    \if@nameauth@BigLock\@nameauth@Lockfalse\fi
266    \if@nameauth@Lock
267      \let\ex\expandafter%
```

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

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

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

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

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

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

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

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

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

```
281        \if@nameauth@DoCaps
282          \let\carga\arga%
283          \let\crootb\rootb%
284          \let\csuffb\suffb%
285          \let\cargc\argc%
286          \unless\if@nameauth@AltFormat
```

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

```
287            \unless\ifx\arga\@empty
288              \def\test{#1}%
289              \ex\@nameauth@UTFtest\ex{\test}%
290              \if@nameauth@UTF
291                \ex\def\ex\carga\ex{\ex\@nameauth@CapUTF\ex{\test}}%
292              \else
293                \ex\def\ex\carga\ex{\ex\@nameauth@Cap\ex{\test}}%
294              \fi
295            \fi
```

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

```
296            \def\test{#2}%
297            \ex\@nameauth@UTFtest\ex{\test}%
298            \if@nameauth@UTF
299              \ex\def\ex\crootb\ex{\ex\@nameauth@CapUTF\ex{\rootb}}%
300            \else
301              \ex\def\ex\crootb\ex{\ex\@nameauth@Cap\ex{\rootb}}%
302            \fi
```

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

```
303            \unless\ifx\suffb\@empty
304              \def\test{#2}%
305              \ex\@nameauth@UTFtestS\ex{\test}%
306              \protected@edef\test{\@nameauth@GetSuff{#2}}%
307              \if@nameauth@UTF
308                \protected@edef\test{\@nameauth@Suffix{#2}}%
309                \ex\def\ex\csuffb\ex{\ex\@nameauth@CapUTF\ex{\test}}%
310              \else
311                \edef\test{\@nameauth@Suffix{#2}}%
312                \ex\def\ex\csuffb\ex{\ex\@nameauth@Cap\ex{\test}}%
313              \fi
314            \fi
```

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

```
315            \unless\ifx\argc\@empty
```

```
316          \def\test{#3}%
317          \ex\@nameauth@UTFtest\ex{\test}%
318          \if@nameauth@UTF
319            \ex\def\ex\cargc\ex{\ex\@nameauth@CapUTF\ex{\test}}%
320          \else
321            \ex\def\ex\cargc\ex{\ex\@nameauth@Cap\ex{\test}}%
322          \fi
323        \fi
324      \fi
325      \let\arga\carga%
326      \let\rootb\crootb%
327      \let\suffb\csuffb%
328      \let\argc\cargc%
329    \fi
```

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

```
330      \if@nameauth@AllThis
331        \protected@edef\rootb{\MakeUppercase{\@nameauth@Root{#2}}}%
332      \fi
```

Use non-breaking spaces and commas as desired.

```
333      \edef\Space{\space}%
334      \edef\SpaceX{\space}%
335      \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
336      \if@nameauth@NBSPX\edef\SpaceX{\nobreakspace}\fi
337      \unless\ifx\arga\@empty
338        \if@nameauth@AlwaysComma
339          \edef\Space{,\space}%
340          \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
341        \fi
342        \if@nameauth@ShowComma
343          \edef\Space{,\space}%
344          \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
345        \fi
346        \if@nameauth@NoComma
347          \edef\Space{\space}%
348          \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
349        \fi
350      \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.

```
351      \let\SNN\rootb%
352      \ifx\arga\@empty
353        \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.

```
354          \let\FNN\suffb%
355          \let\SNN\rootb%
356          \@nameauth@NonWest{\csb#4}%
357        \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.

```
358        \ifx\suffb\@empty
359          \let\FNN\argc%
360          \let\SNN\rootb%
```

```
361            \@nameauth@NonWest{\csbc#4}%
362          \else
363            \let\FNN\argc%
364            \let\SNN\rootb%
365            \@nameauth@NonWest{\csb#4}%
366          \fi
367        \fi
368      \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.

```
369        \ifx\argc\@empty
370          \let\FNN\arga%
371        \else
372          \let\FNN\argc%
373        \fi
374        \unless\ifx\suffb\@empty
375          \def\SNN{\rootb\Space\suffb}%
376          \if@nameauth@ShortSNN\let\SNN\rootb\fi
377        \fi
378        \@nameauth@West{\csab#4}%
379      \fi
380    \fi
381 }
```

`\@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.

```
382 \newcommand*\@nameauth@NonWest[1]
383 {%
384   \if@nameauth@BigLock\@nameauth@Lockfalse\fi
385   \if@nameauth@Lock
386     \unless\ifcsname#1\endcsname
387       \@nameauth@FirstFormattrue%
388     \fi
389     \if@nameauth@InAKA
390       \if@nameauth@AltAKA
391         \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi
392         \@nameauth@FullNamefalse%
393         \@nameauth@FirstNametrue%
394       \else
395         \@nameauth@FullNametrue%
396         \@nameauth@FirstNamefalse%
397       \fi
398     \else
399       \unless\ifcsname#1\endcsname
400         \@nameauth@FullNametrue%
401         \@nameauth@FirstNamefalse%
402       \fi
403     \fi
404     \if@nameauth@FirstName
405       \@nameauth@FullNamefalse%
406     \fi
407     \ifx\FNN\@empty
408       \@nameauth@Hook{\SNN}%
409     \else
410       \if@nameauth@FullName
```

```
411        \if@nameauth@RevThis
412          \@nameauth@Hook{\FNN\Space\SNN}%
413        \else
414          \@nameauth@Hook{\SNN\Space\FNN}%
415        \fi
416      \else
417        \if@nameauth@FirstName
418          \if@nameauth@EastFN
419            \@nameauth@Hook{\FNN}%
420          \else
421            \@nameauth@Hook{\SNN}%
422          \fi
423        \else
424          \@nameauth@Hook{\SNN}%
425        \fi
426      \fi
427    \fi
428    \unless\ifcsname#1\endcsname
429      \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
430    \fi
```

We have to reset these flags here because both the naming and cross-referencing macros use the parser.

```
431      \if@nameauth@OldReset
432        \@nameauth@FullNamefalse%
433        \@nameauth@FirstNamefalse%
434      \else
435        \global\@nameauth@FullNamefalse%
436        \global\@nameauth@FirstNamefalse%
437      \fi
438    \fi
439 }
```

\@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.

```
440 \newcommand*\@nameauth@West[1]
441 {%
442    \if@nameauth@BigLock\@nameauth@Lockfalse\fi
443    \if@nameauth@Lock
444      \unless\ifcsname#1\endcsname
445        \@nameauth@FirstFormattrue%
446      \fi
447      \if@nameauth@InAKA
448        \if@nameauth@AltAKA
449          \@nameauth@FullNamefalse%
450          \@nameauth@FirstNametrue%
451        \else
452          \@nameauth@FullNametrue%
453          \@nameauth@FirstNamefalse%
454        \fi
455      \else
456        \unless\ifcsname#1\endcsname
457          \@nameauth@FullNametrue%
458          \@nameauth@FirstNamefalse%
459        \fi
460      \fi
461      \if@nameauth@FirstName
```

```
462        \@nameauth@FullNamefalse%
463      \fi
464      \if@nameauth@FullName
465        \if@nameauth@RevThis
466          \@nameauth@Hook{\SNN\SpaceX\FNN}%
467        \else
468          \if@nameauth@RevThisComma
469            \edef\RevSpace{,\SpaceX}%
470            \@nameauth@Hook{\SNN\RevSpace\FNN}%
471          \else
472            \@nameauth@Hook{\FNN\SpaceX\SNN}%
473          \fi
474        \fi
475      \else
476        \if@nameauth@FirstName
477          \@nameauth@Hook{\FNN}%
478        \else
479          \@nameauth@Hook{\rootb}%
480        \fi
481      \fi
482      \unless\ifcsname#1\endcsname
483        \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
484      \fi
```

We have to reset these flags here because both the naming and cross-referencing macros use the parser.

```
485      \if@nameauth@OldReset
486        \@nameauth@FullNamefalse%
487        \@nameauth@FirstNamefalse%
488      \else
489        \global\@nameauth@FullNamefalse%
490        \global\@nameauth@FirstNamefalse%
491      \fi
492    \fi
493 }
```

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.

```
494 \newcommand*\@nameauth@Hook[1]
495 {%
496    \if@nameauth@BigLock\@nameauth@Lockfalse\fi
497    \if@nameauth@Lock
498      \@nameauth@InHooktrue%
499      \protected@edef\test{#1}%
500      \expandafter\@nameauth@TestDot\expandafter{\test}%
501      \if@nameauth@InAKA
502        \if@nameauth@AlwaysFormat
503          \@nameauth@FirstFormattrue%
504        \else
505          \unless\if@nameauth@AKAFormat
506          \@nameauth@FirstFormatfalse\fi
507        \fi
508        \if@nameauth@MainFormat
509          \if@nameauth@FirstFormat
```

87

```
510        \bgroup\NamesFormat{#1}\egroup%
511      \else
512        \bgroup\MainNameHook{#1}\egroup%
513      \fi
514    \else
515      \if@nameauth@FirstFormat
516        \bgroup\FrontNamesFormat{#1}\egroup%
517      \else
518        \bgroup\FrontNameHook{#1}\egroup%
519      \fi
520    \fi
521  \else
522    \if@nameauth@AlwaysFormat
523      \@nameauth@FirstFormattrue%
524    \fi
525    \if@nameauth@MainFormat
526      \if@nameauth@FirstFormat
527        \bgroup\NamesFormat{#1}\egroup%
528      \else
529        \bgroup\MainNameHook{#1}\egroup%
530      \fi
531    \else
532      \if@nameauth@FirstFormat
533        \bgroup\FrontNamesFormat{#1}\egroup%
534      \else
535        \bgroup\FrontNameHook{#1}\egroup%
536      \fi
537    \fi
538  \fi
```

We have to reset this flag here because both the naming and cross-referencing macros use the parser.

```
539      \if@nameauth@OldReset
540        \@nameauth@FirstFormatfalse%
541      \else
542        \global\@nameauth@FirstFormatfalse%
543      \fi
544      \@nameauth@InHookfalse%
545    \fi
546 }
```

Indexing Internals

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

```
547 \newcommand*\@nameauth@Index[2]
548 {%
549   \let\ex\expandafter%
550   \if@nameauth@DoIndex
551     \ifcsname#1!TAG\endcsname
552       \protected@edef\Tag{\csname#1!TAG\endcsname}%
553       \ex\def\ex\ShortTag\ex{\ex\@nameauth@TrimTag\ex{\Tag}}%
554       \ifcsname#1!PRE\endcsname
555         \protected@edef\Pre{\csname#1!PRE\endcsname}%
556         \if@nameauth@Xref
557           \protected@edef\Entry{\Pre#2\ShortTag}%
558         \else
```

```
559            \protected@edef\Entry{\Pre#2\Tag}%
560          \fi
561        \else
562          \if@nameauth@Xref
563            \protected@edef\Entry{#2\ShortTag}%
564          \else
565            \protected@edef\Entry{#2\Tag}%
566          \fi
567        \fi
568      \else
569        \ifcsname#1!PRE\endcsname
570          \protected@edef\Pre{\csname#1!PRE\endcsname}%
571          \protected@edef\Entry{\Pre#2}%
572        \else
573          \protected@edef\Entry{#2}%
574        \fi
575      \fi
576      \ex\index\ex{\Entry}%
577    \fi
578 }
```

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

```
579 \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.

```
580 \newcommandx*\@nameauth@Debug[3][1=\@empty, 3=\@empty]
581 {%
582   \bgroup%
583     \def\index##1{##1}%
584     \@nameauth@DoIndextrue%
585     \protected@edef\arga{\trim@spaces{#1}}%
586     \protected@edef\argc{\trim@spaces{#3}}%
587     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
588     \def\csb{\@nameauth@Clean{#2}}%
589     \def\csbc{\@nameauth@Clean{#2,#3}}%
590     \def\csab{\@nameauth@Clean{#1!#2}}%
591     \@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.

```
592     \ifx\arga\@empty
593       \ifx\argc\@empty
594         \ifdefined\ShortIdxEntry
595           \csundef{\csb!PRE}%
596           \csundef{\csb!TAG}%
597           \csundef{\csb!PN}%
598           \IndexName[#1]{#2}[#3]%
599         \else
600           \ifdefined\LongIdxEntry
```

89

```
601            \csundef{\csb!PN}%
602            \IndexName[#1]{#2}[#3]%
603          \else
604            \csb%
605          \fi
606        \fi
607      \else
608        \ifx\suffb\@empty
609          \ifdefined\ShortIdxEntry
610            \csundef{\csbc!PRE}%
611            \csundef{\csbc!TAG}%
612            \csundef{\csbc!PN}%
613            \IndexName[#1]{#2}[#3]%
614          \else
615            \ifdefined\LongIdxEntry
616              \csundef{\csbc!PN}%
617              \IndexName[#1]{#2}[#3]%
618            \else
619              \csbc%
620            \fi
621          \fi
622        \else
623          \ifdefined\ShortIdxEntry
624            \csundef{\csb!PRE}%
625            \csundef{\csb!TAG}%
626            \csundef{\csb!PN}%
627            \IndexName[#1]{#2}[#3]%
628          \else
629            \ifdefined\LongIdxEntry
630              \csundef{\csb!PN}%
631              \IndexName[#1]{#2}[#3]%
632            \else
633              \csb%
634            \fi
635          \fi
636        \fi
637      \fi
638    \else
639      \ifdefined\ShortIdxEntry
640        \csundef{\csab!PRE}%
641        \csundef{\csab!TAG}%
642        \csundef{\csab!PN}%
643        \IndexName[#1]{#2}[#3]%
644      \else
645        \ifdefined\LongIdxEntry
646          \csundef{\csab!PN}%
647          \IndexName[#1]{#2}[#3]%
648        \else
649          \csab%
650        \fi
651      \fi
652    \fi
653    \global\undef{\LongIdxEntry}%
654    \global\undef{\ShortIdxEntry}%
655  \egroup%
656 }
```

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.

```
657 \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.

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

\CapName   Capitalize entire required name. Overrides \CapThis for surnames.

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

\AllCapsInactive   Turn off global surname capitalization.

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

\AllCapsActive   Turn on global surname capitalization. Activates \CapName for every name.

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

Syntactic Formatting — Reversing

\RevName   Reverse name order.

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

\ReverseInactive   Turn off global name reversing.

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

\ReverseActive   Turn on global name reversing. Activates \RevName for every name.

```
665 \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*⟩.

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

Syntactic Formatting — Reversing with Commas

\RevComma   Last name, comma, first name.

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

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

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

\ReverseCommaActive   Turn on global "last-name-comma-first." Activates \RevComma for every name.

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

Alternate Formatting

\AltFormatActive   Turn on alternate formatting, engage the formatting macros.

```
673 \newcommand*\AltFormatActive{%
674    \global\@nameauth@AltFormattrue%
675    \global\@nameauth@DoAlttrue%
676 }
```

\AltFormatActive*	Turn on alternate formatting, disengage the formatting macros.

```
677 \WithSuffix{\newcommand*}\AltFormatActive*{%
678   \global\@nameauth@AltFormattrue%
679   \global\@nameauth@DoAltfalse%
680 }
```

\AltFormatInactive	Turn off alternate formatting altogether.

```
681 \newcommand*\AltFormatInactive{%
682   \global\@nameauth@AltFormatfalse%
683   \global\@nameauth@DoAltfalse%
684 }
```

\AltOn	Locally turn on alternate formatting.

```
685 \newcommand*\AltOn{%
686   \if@nameauth@InHook
687     \if@nameauth@AltFormat\@nameauth@DoAlttrue\fi
688   \fi
689 }
```

\AltOff	Locally turn off alternate formatting.

```
690 \newcommand*\AltOff{%
691   \if@nameauth@InHook
692     \if@nameauth@AltFormat\@nameauth@DoAltfalse\fi
693   \fi
694 }
```

\AltCaps	Alternate discretionary capping macro triggered by \CapThis.

```
695 \newcommand*\AltCaps[1]{%
696   \if@nameauth@InHook
697     \if@nameauth@DoCaps\MakeUppercase{#1}\else#1\fi
698   \else#1%
699   \fi
700 }
```

\textSC	Alternate formatting macro: small caps when active.

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

\textUC	Alternate formatting macro: uppercase when active.

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

\textIT	Alternate formatting macro: italic when active.

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

\textBF	Alternate formatting macro: boldface when active.

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

Syntactic Formatting — Affixes

\ShowComma	Put comma between name and suffix one time.

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

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

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

\DropAffix	Suppress the affix in a long Western name.

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

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

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

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

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

Post-Processing — Main Versus Front Matter

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

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

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

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

Name Decisions — First / Subsequent Reference

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

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

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

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

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

718 `\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.

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

\GlobalNames	`\GlobalNames` sets `@nameauth@LocalNames` false. This restores the default behavior of `\ForgetName` and `\SubvertName`.

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

Index Operations

\IndexInactive	Turn off global indexing of names.

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

\SkipIndex	Turn off the next instance of indexing in `\Name`, `\FName`, and starred forms.

722 `\newcommand*\SkipIndex{\@nameauth@SkipIndextrue}`

\JustIndex	Makes the next call to `\Name`, `\FName`, and starred forms act like `\IndexName`. Overrides `\SkipIndex`.

723 `\newcommand*\JustIndex{\@nameauth@JustIndextrue}`

\IndexActive	Turn on global indexing of names.

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

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

```
725 \newcommand*\IndexActual[1]
726   {\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!

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

3.6 User Interface Macros: General

\ShowPattern   This displays the pattern that the name arguments generate; maybe useful for debugging.

```
728 \newcommand*\ShowPattern{\@nameauth@Debug}
```

\ShowIdxPageref   This displays the index entry that will be generated. This may be useful for debugging.

```
729 \newcommand*\ShowIdxPageref%
730   {\def\LongIdxEntry{}\ShowPattern}
```

\ShowIdxPageref*   This displays an index entry with no tag. This may be useful for debugging.

```
731 \WithSuffix{\newcommand*}\ShowIdxPageref*%
732   {\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.

```
733 \newcommand*\NameParser
734 {%
735   \if@nameauth@InHook
736     \let\SNN\rootb%
737     \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.

```
738       \ifx\argc\@empty
739         \let\FNN\suffb%
740       \else
741         \let\FNN\argc%
742       \fi
743       \ifx\suffb\@empty
```

Mononym case

```
744         \ifx\FNN\@empty
745           \SNN%
746         \else
```

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

```
747           \if@nameauth@FullName%
748             \if@nameauth@RevThis
749               \FNN\Space\SNN%
750             \else
751               \SNN\Space\FNN%
752             \fi
753           \else
754             \if@nameauth@FirstName
755               \if@nameauth@EastFN
756                 \FNN%
757               \else
758                 \SNN%
759               \fi
```

94

```
760                \else
761                    \SNN%
762                \fi
763            \fi
764        \fi
765    \else
```

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

```
766            \if@nameauth@FullName
767                \if@nameauth@RevThis
768                    \FNN\Space\SNN%
769                \else
770                    \SNN\Space\FNN%
771                \fi
772            \else
773                \if@nameauth@FirstName
774                    \if@nameauth@EastFN
775                        \FNN%
776                    \else
777                        \SNN%
778                    \fi
779                \else
780                    \SNN%
781                \fi
782            \fi
783        \fi
784    \else
```

Western name with name reversing and suffixes

```
785        \ifx\argc\@empty
786            \let\FNN\arga%
787        \else
788            \let\FNN\argc%
789        \fi
790        \unless\ifx\suffb\@empty
791            \def\SNN{\rootb\Space\suffb}%
792            \if@nameauth@ShortSNN\let\SNN\rootb\fi%
793        \fi
794        \if@nameauth@FullName
795            \if@nameauth@RevThis
796                \SNN\SpaceX\FNN%
797            \else
798                \if@nameauth@RevThisComma
799                    \SNN\RevSpace\FNN%
800                \else
801                    \FNN\SpaceX\SNN%
802                \fi
803            \fi
804        \else
805            \if@nameauth@FirstName
806                \FNN%
807            \else
808                \let\SNN\rootb%
809                \SNN%
810            \fi
811        \fi
812    \fi
```

Traditional Naming Interface

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

815 \newcommand\Name{\NameauthName}

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

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

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

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

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

819 \WithSuffix{\newcommand*}\FName*%
820   {\@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.

821 \newcommand*\IndexProtect
822 {%
823   \@nameauth@DoIndexfalse%
824   \@nameauth@BigLocktrue%
825 }

\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.

826 \newcommandx*\IndexName[3][1=\@empty, 3=\@empty]
827 {%
828   \protected@edef\arga{\trim@spaces{#1}}%
829   \protected@edef\rootb{\@nameauth@Root{#2}}%
830   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
831   \protected@edef\argc{\trim@spaces{#3}}%
832   \def\csb{\@nameauth@Clean{#2}}%
833   \def\csbc{\@nameauth@Clean{#2,#3}}%
834   \def\csab{\@nameauth@Clean{#1!#2}}%

Test for malformed input.

835   \@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.

836   \ifx\arga\@empty
837   \ifx\argc\@empty
838   \ifcsname\csb!PN\endcsname
839   \if@nameauth@Verbose
840   \PackageWarning{nameauth}%
841   {macro \IndexName: XRef: #2 exists}%
842   \fi
843   \else
844   \ifx\suffb\@empty

96

mononym or Eastern / ancient name, new syntax

```
845              \@nameauth@Index{\csb}{\rootb}%
846            \else
847              \@nameauth@Index{\csb}{\rootb\space\suffb}%
848            \fi
849          \fi
850        \else
851          \ifx\suffb\@empty
852            \ifcsname\csbc!PN\endcsname
853              \if@nameauth@Verbose
854                \PackageWarning{nameauth}%
855                {macro \IndexName: XRef: #2 #3 exists}%
856              \fi
857            \else
```

Eastern or ancient name, older syntax

```
858              \@nameauth@Index{\csbc}{\rootb\space\argc}%
859            \fi
860          \else
861            \ifcsname\csb!PN\endcsname
862              \if@nameauth@Verbose
863                \PackageWarning{nameauth}%
864                {macro \IndexName: XRef: #2 exists}%
865              \fi
866            \else
```

Eastern or ancient name, new syntax, alternate name ignored

```
867              \@nameauth@Index{\csb}{\rootb\space\suffb}%
868            \fi
869          \fi
870        \fi
871    \else
872      \ifcsname\csab!PN\endcsname
873        \if@nameauth@Verbose
874          \PackageWarning{nameauth}%
875          {macro \IndexName: XRef: #1 #2 exists}%
876        \fi
877      \else
```

Western name, without and with affix

```
878        \ifx\suffb\@empty
879          \@nameauth@Index{\csab}%
880          {\rootb,\space\arga}%
881        \else
882          \@nameauth@Index{\csab}%
883          {\rootb,\space\arga,\space\suffb}%
884        \fi
885      \fi
886    \fi
887 }
```

\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.

```
888 \newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
889 {%
890   \protected@edef\arga{\trim@spaces{#1}}%
891   \protected@edef\rootb{\@nameauth@Root{#2}}%
```

```
892    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
893    \protected@edef\argc{\trim@spaces{#3}}%
894    \protected@edef\target{#4}%
895    \def\csb{\@nameauth@Clean{#2}}%
896    \def\csbc{\@nameauth@Clean{#2,#3}}%
897    \def\csab{\@nameauth@Clean{#1!#2}}%
898    \let\ex\expandafter%
```

Test for malformed input.

```
899    \@nameauth@Error{#2}{macro \string\IndexRef}%
900    \@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.

```
901    \ifx\arga\@empty
902      \ifx\argc\@empty
903        \ifcsname\csb!PN\endcsname
904          \if@nameauth@Verbose
905            \PackageWarning{nameauth}%
906            {macro \IndexRef: XRef: #2 exists}%
907          \fi
908        \else
909          \ifx\suffb\@empty
```

mononym or Eastern / ancient name, new syntax

```
910            \if@nameauth@SeeAlso
911              \@nameauth@Index{\csb}{\rootb|seealso{\target}}%
912            \else
913              \@nameauth@Index{\csb}{\rootb|see{\target}}%
914            \fi
915          \else
916            \if@nameauth@SeeAlso
917              \@nameauth@Index{\csb}%
918                {\rootb\space\suffb|seealso{\target}}%
919            \else
920              \@nameauth@Index{\csb}%
921                {\rootb\space\suffb|see{\target}}%
922            \fi
923          \fi
924          \csgdef{\csb!PN}{}%
925        \fi
926      \else
927        \ifx\suffb\@empty
928          \ifcsname\csbc!PN\endcsname
929            \if@nameauth@Verbose
930              \PackageWarning{nameauth}%
931              {macro \IndexRef: XRef: #2 #3 exists}%
932            \fi
933          \else
```

Eastern or ancient name, older syntax

```
934            \if@nameauth@SeeAlso
935              \@nameauth@Index{\csbc}%
936                {\rootb\space\argc|seealso{\target}}%
937            \else
938              \@nameauth@Index{\csbc}%
939                {\rootb\space\argc|see{\target}}%
940            \fi
```

```
941          \csgdef{\csbc!PN}{}%
942        \fi
943      \else
944        \ifcsname\csb!PN\endcsname
945          \if@nameauth@Verbose
946            \PackageWarning{nameauth}%
947            {macro \IndexRef: XRef: #2 exists}%
948          \fi
949        \else
```

Eastern or ancient name, new syntax, alternate name ignored

```
950            \if@nameauth@SeeAlso
951              \@nameauth@Index{\csb}%
952                {\rootb\space\suffb|seealso{\target}}%
953            \else
954              \@nameauth@Index{\csb}%
955                {\rootb\space\suffb|see{\target}}%
956            \fi
957            \csgdef{\csb!PN}{}%
958        \fi
959      \fi
960    \fi
961  \else
962    \ifcsname\csab!PN\endcsname
963      \if@nameauth@Verbose
964        \PackageWarning{nameauth}%
965        {macro \IndexRef: XRef: #1 #2 exists}%
966      \fi
967    \else
```

Western name, without and with affix

```
968        \ifx\suffb\@empty
969          \if@nameauth@SeeAlso
970            \@nameauth@Index{\csab}%
971            {\rootb,\space\arga|seealso{\target}}%
972          \else
973            \@nameauth@Index{\csab}%
974            {\rootb,\space\arga|see{\target}}%
975          \fi
976        \else
977          \if@nameauth@SeeAlso
978            \@nameauth@Index{\csab}%
979            {\rootb,\space\arga,\space\suffb|seealso{\target}}%
980          \else
981            \@nameauth@Index{\csab}%
982            {\rootb,\space\arga,\space\suffb|see{\target}}%
983          \fi
984        \fi
985        \csgdef{\csab!PN}{}%
986    \fi
987  \fi
988  \@nameauth@Xreffalse%
```

This may not be necessary, but we do it for consistency.

```
989  \if@nameauth@OldReset
990    \@nameauth@SeeAlsofalse%
991  \else
992    \global\@nameauth@SeeAlsofalse%
```

```
993        \fi
994 }
```

\ExcludeName   This macro prevents a name from being indexed.

```
995 \newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]
996 {%
997     \protected@edef\arga{\trim@spaces{#1}}%
998     \protected@edef\argc{\trim@spaces{#3}}%
999     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1000    \def\csb{\@nameauth@Clean{#2}}%
1001    \def\csbc{\@nameauth@Clean{#2,#3}}%
1002    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1003    \@nameauth@Error{#2}{macro \string\ExcludeName}%
1004    \ifx\arga\@empty
1005      \ifx\argc\@empty
1006        \if@nameauth@Verbose
1007          \ifcsname\csb!MN\endcsname
1008            \PackageWarning{nameauth}%
1009            {macro \ExcludeName: Reference: #2 exists}%
1010          \fi
1011          \ifcsname\csb!NF\endcsname
1012            \PackageWarning{nameauth}%
1013            {macro \ExcludeName: Reference: #2 exists}%
1014          \fi
1015        \fi
1016        \ifcsname\csb!PN\endcsname
1017          \if@nameauth@Verbose
1018            \PackageWarning{nameauth}%
1019            {macro \ExcludeName: Xref: #2 exists}%
1020          \fi
1021        \else
1022          \csgdef{\csb!PN}{!}%
1023        \fi
1024      \else
1025        \ifx\suffb\@empty
1026          \if@nameauth@Verbose
1027            \ifcsname\csbc!MN\endcsname
1028              \PackageWarning{nameauth}%
1029              {macro \ExcludeName: Reference: #2 #3 exists}%
1030            \fi
1031            \ifcsname\csbc!NF\endcsname
1032              \PackageWarning{nameauth}%
1033              {macro \ExcludeName: Reference: #2 #3 exists}%
1034            \fi
1035          \fi
1036          \csgdef{\csbc!PN}{!}%
1037          \ifcsname\csbc!PN\endcsname
1038            \if@nameauth@Verbose
1039              \PackageWarning{nameauth}%
1040              {macro \ExcludeName: Xref: #2 #3 exists}%
1041            \fi
1042          \else
1043            \csgdef{\csbc!PN}{!}%
1044          \fi
1045        \else
```

```
1046          \if@nameauth@Verbose
1047            \ifcsname\csb!MN\endcsname
1048              \PackageWarning{nameauth}%
1049              {macro \ExcludeName: Reference: #2 exists}%
1050            \fi
1051            \ifcsname\csb!NF\endcsname
1052              \PackageWarning{nameauth}%
1053              {macro \ExcludeName: Reference: #2 exists}%
1054            \fi
1055          \fi
1056          \ifcsname\csb!PN\endcsname
1057            \if@nameauth@Verbose
1058              \PackageWarning{nameauth}%
1059              {macro \ExcludeName: Xref: #2 exists}%
1060            \fi
1061          \else
1062            \csgdef{\csb!PN}{!}%
1063          \fi
1064        \fi
1065      \fi
1066    \else
1067      \if@nameauth@Verbose
1068        \ifcsname\csab!MN\endcsname
1069          \PackageWarning{nameauth}%
1070          {macro \ExcludeName: Reference: #1 #2 exists}%
1071        \fi
1072        \ifcsname\csab!NF\endcsname
1073          \PackageWarning{nameauth}%
1074          {macro \ExcludeName: Reference: #1 #2 exists}%
1075        \fi
1076      \fi
1077      \ifcsname\csab!PN\endcsname
1078        \if@nameauth@Verbose
1079          \PackageWarning{nameauth}%
1080          {macro \ExcludeName: Xref: #1 #2 exists}%
1081        \fi
1082      \else
1083        \csgdef{\csab!PN}{!}%
1084      \fi
1085    \fi
1086 }
```

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

```
1087 \newcommandx*\IncludeName[3][1=\@empty, 3=\@empty]
1088 {%
1089   \protected@edef\arga{\trim@spaces{#1}}%
1090   \protected@edef\argc{\trim@spaces{#3}}%
1091   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1092   \def\csb{\@nameauth@Clean{#2}}%
1093   \def\csbc{\@nameauth@Clean{#2,#3}}%
1094   \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and undefine only an "excluded" name.

```
1095   \@nameauth@Error{#2}{macro \string\IncludeName}%
1096   \ifx\arga\@empty
1097     \ifx\argc\@empty
1098       \ifcsname\csb!PN\endcsname
```

```
1099            \edef\testex{\csname\csb!PN\endcsname}%
1100            \unless\ifx\testex\@empty\global\csundef{\csb!PN}%
1101            \else
1102              \if@nameauth@Verbose
1103                \PackageWarning{nameauth}%
1104                {macro \IncludeName: Xref: #2 exists}%
1105              \fi
1106            \fi
1107          \fi
1108        \else
1109          \ifx\suffb\@empty
1110            \ifcsname\csbc!PN\endcsname
1111              \edef\testex{\csname\csbc!PN\endcsname}%
1112              \unless\ifx\testex\@empty\global\csundef{\csbc!PN}%
1113              \else
1114                \if@nameauth@Verbose
1115                  \PackageWarning{nameauth}%
1116                  {macro \IncludeName: Xref: #2 #3 exists}%
1117                \fi
1118              \fi
1119            \fi
1120          \else
1121            \ifcsname\csb!PN\endcsname
1122              \edef\testex{\csname\csb!PN\endcsname}%
1123              \unless\ifx\testex\@empty\global\csundef{\csb!PN}%
1124              \else
1125                \if@nameauth@Verbose
1126                  \PackageWarning{nameauth}%
1127                  {macro \IncludeName: Xref: #2 exists}%
1128                \fi
1129              \fi
1130            \fi
1131          \fi
1132        \fi
1133      \else
1134        \ifcsname\csab!PN\endcsname
1135          \edef\testex{\csname\csab!PN\endcsname}%
1136          \unless\ifx\testex\@empty\global\csundef{\csab!PN}%
1137          \else
1138            \if@nameauth@Verbose
1139              \PackageWarning{nameauth}%
1140              {macro \IncludeName: Xref: #1 #2 exists}%
1141            \fi
1142          \fi
1143        \fi
1144    \fi
1145 }
```

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

```
1146 \WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
1147 {%
1148    \protected@edef\arga{\trim@spaces{#1}}%
1149    \protected@edef\argc{\trim@spaces{#3}}%
1150    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1151    \def\csb{\@nameauth@Clean{#2}}%
1152    \def\csbc{\@nameauth@Clean{#2,#3}}%
```

```
1153    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1154    \@nameauth@Error{#2}{macro \string\IncludeName*}%
1155    \ifx\arga\@empty
1156      \ifx\argc\@empty
1157        \global\csundef{\csb!PN}%
1158      \else
1159        \ifx\suffb\@empty
1160          \global\csundef{\csbc!PN}%
1161        \else
1162          \global\csundef{\csb!PN}%
1163        \fi
1164      \fi
1165    \else
1166      \global\csundef{\csab!PN}%
1167    \fi
1168 }
```

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

```
1169 \newcommandx*\PretagName[4][1=\@empty, 3=\@empty]
1170 {%
1171    \protected@edef\arga{\trim@spaces{#1}}%
1172    \protected@edef\argc{\trim@spaces{#3}}%
1173    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1174    \def\csb{\@nameauth@Clean{#2}}%
1175    \def\csbc{\@nameauth@Clean{#2,#3}}%
1176    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1177    \@nameauth@Error{#2}{macro \string\PretagName}%
1178    \ifx\arga\@empty
1179      \ifx\argc\@empty
1180        \ifcsname\csb!PN\endcsname
1181          \if@nameauth@Verbose
1182            \PackageWarning{nameauth}%
1183            {macro \PretagName: tagging xref: #2}%
1184          \fi
1185        \fi
1186        \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
1187      \else
1188        \ifx\suffb\@empty
1189          \ifcsname\csbc!PN\endcsname
1190            \if@nameauth@Verbose
1191              \PackageWarning{nameauth}%
1192              {macro \PretagName: tagging xref: #2 #3}%
1193            \fi
1194          \fi
1195          \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
1196        \else
1197          \ifcsname\csb!PN\endcsname
1198            \if@nameauth@Verbose
1199              \PackageWarning{nameauth}%
1200              {macro \PretagName: tagging xref: #2}%
1201            \fi
1202          \fi
```

```
1203            \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
1204          \fi
1205        \fi
1206     \else
1207       \ifcsname\csab!PN\endcsname
1208          \if@nameauth@Verbose
1209            \PackageWarning{nameauth}%
1210            {macro \PretagName: tagging xref: #1 #2}%
1211          \fi
1212        \fi
1213        \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
1214     \fi
1215 }
```

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

```
1216 \newcommandx*\TagName[4][1=\@empty, 3=\@empty]
1217 {%
1218     \protected@edef\arga{\trim@spaces{#1}}%
1219     \protected@edef\argc{\trim@spaces{#3}}%
1220     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1221     \def\csb{\@nameauth@Clean{#2}}%
1222     \def\csbc{\@nameauth@Clean{#2,#3}}%
1223     \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, defining the macros used by \@nameauth@Index.

```
1224     \@nameauth@Error{#2}{macro \string\TagName}%
1225     \ifx\arga\@empty
1226        \ifx\argc\@empty
1227           \ifcsname\csb!PN\endcsname
1228             \if@nameauth@Verbose
1229               \PackageWarning{nameauth}%
1230               {macro \TagName: not tagging xref: #2}%
1231             \fi
1232           \else
1233             \csgdef{\csb!TAG}{#4}%
1234           \fi
1235        \else
1236           \ifx\suffb\@empty
1237             \ifcsname\csbc!PN\endcsname
1238               \if@nameauth@Verbose
1239                 \PackageWarning{nameauth}%
1240                 {macro \TagName: not tagging xref: #2 #3}%
1241               \fi
1242             \else
1243               \csgdef{\csbc!TAG}{#4}%
1244             \fi
1245           \else
1246             \ifcsname\csb!PN\endcsname
1247               \if@nameauth@Verbose
1248                 \PackageWarning{nameauth}%
1249                 {macro \TagName: not tagging xref: #2}%
1250               \fi
1251             \else
1252               \csgdef{\csb!TAG}{#4}%
1253             \fi
1254           \fi
1255        \fi
```

```
1256    \else
1257      \ifcsname\csab!PN\endcsname
1258        \if@nameauth@Verbose
1259          \PackageWarning{nameauth}%
1260          {macro \TagName: not tagging xref: #1 #2}%
1261        \fi
1262      \else
1263        \csgdef{\csab!TAG}{#4}%
1264      \fi
1265    \fi
1266 }
```

\UntagName   This deletes an index tag.

```
1267 \newcommandx*\UntagName[3][1=\@empty, 3=\@empty]
1268 {%
1269    \protected@edef\arga{\trim@spaces{#1}}%
1270    \protected@edef\argc{\trim@spaces{#3}}%
1271    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1272    \def\csb{\@nameauth@Clean{#2}}%
1273    \def\csbc{\@nameauth@Clean{#2,#3}}%
1274    \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, undefining the index tag macros.

```
1275    \@nameauth@Error{#2}{macro \string\UntagName}%
1276    \ifx\arga\@empty
1277      \ifx\argc\@empty
1278        \global\csundef{\csb!TAG}%
1279      \else
1280        \ifx\suffb\@empty
1281          \global\csundef{\csbc!TAG}%
1282        \else
1283          \global\csundef{\csb!TAG}%
1284        \fi
1285      \fi
1286    \else
1287      \global\csundef{\csab!TAG}%
1288    \fi
1289 }
```

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.

```
1290 \newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
1291 {%
1292    \protected@edef\arga{\trim@spaces{#1}}%
1293    \protected@edef\argc{\trim@spaces{#3}}%
1294    \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1295    \def\csb{\@nameauth@Clean{#2}}%
1296    \def\csbc{\@nameauth@Clean{#2,#3}}%
1297    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1298    \@nameauth@Error{#2}{macro \string\NameAddInfo}%
1299    \ifx\arga\@empty
1300      \ifx\argc\@empty
1301        \csgdef{\csb!DB}{#4}%
```

```
1302        \else
1303          \ifx\Suff\@empty
1304            \csgdef{\csbc!DB}{#4}%
1305          \else
1306            \csgdef{\csb!DB}{#4}%
1307          \fi
1308        \fi
1309      \else
1310        \csgdef{\csab!DB}{#4}%
1311      \fi
1312 }
```

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

```
1313 \newcommandx*\NameQueryInfo[3][1=\@empty, 3=\@empty]
1314 {%
1315    \protected@edef\arga{\trim@spaces{#1}}%
1316    \protected@edef\argc{\trim@spaces{#3}}%
1317    \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1318    \def\csb{\@nameauth@Clean{#2}}%
1319    \def\csbc{\@nameauth@Clean{#2,#3}}%
1320    \def\csab{\@nameauth@Clean{#1!#2}}%
1321    \unless\if@nameauth@BigLock
```

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

```
1322      \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
1323      \ifx\arga\@empty
1324        \ifx\argc\@empty
1325          \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1326        \else
1327          \ifx\Suff\@empty
1328            \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
1329          \else
1330            \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1331          \fi
1332        \fi
1333      \else
1334        \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
1335      \fi
1336    \fi
1337 }
```

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

```
1338 \newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
1339 {%
1340    \protected@edef\arga{\trim@spaces{#1}}%
1341    \protected@edef\argc{\trim@spaces{#3}}%
1342    \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1343    \def\csb{\@nameauth@Clean{#2}}%
1344    \def\csbc{\@nameauth@Clean{#2,#3}}%
1345    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1346    \@nameauth@Error{#2}{macro \string\NameClearInfo}%
1347    \ifx\arga\@empty
1348      \ifx\argc\@empty
1349        \global\csundef{\csb!DB}%
1350      \else
```

```
1351        \ifx\Suff\@empty
1352          \global\csundef{\csbc!DB}%
1353        \else
1354          \global\csundef{\csb!DB}%
1355        \fi
1356      \fi
1357    \else
1358      \global\csundef{\csab!DB}%
1359    \fi
1360 }
```

Name Decisions

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

```
1361 \newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
1362 {%
1363    \protected@edef\arga{\trim@spaces{#1}}%
1364    \protected@edef\argc{\trim@spaces{#3}}%
1365    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1366    \def\csb{\@nameauth@Clean{#2}}%
1367    \def\csbc{\@nameauth@Clean{#2,#3}}%
1368    \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
1369    \@nameauth@Error{#2}{macro \string\IfMainName}%
1370    \ifx\arga\@empty
1371      \ifx\argc\@empty
1372        \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1373      \else
1374        \ifx\suffb\@empty
1375          \ifcsname\csbc!MN\endcsname{#4}\else{#5}\fi
1376        \else
1377          \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1378        \fi
1379      \fi
1380    \else
1381      \ifcsname\csab!MN\endcsname{#4}\else{#5}\fi
1382    \fi
1383 }
```

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

```
1384 \newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
1385 {%
1386    \protected@edef\arga{\trim@spaces{#1}}%
1387    \protected@edef\argc{\trim@spaces{#3}}%
1388    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1389    \def\csb{\@nameauth@Clean{#2}}%
1390    \def\csbc{\@nameauth@Clean{#2,#3}}%
1391    \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
1392    \@nameauth@Error{#2}{macro \string\IfFrontName}%
1393    \ifx\arga\@empty
1394      \ifx\argc\@empty
1395        \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1396      \else
1397        \ifx\suffb\@empty
```

```
1398            \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
1399          \else
1400            \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1401          \fi
1402        \fi
1403      \else
1404        \ifcsname\csab!NF\endcsname{#4}\else{#5}\fi
1405      \fi
1406 }
```

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

```
1407 \newcommandx\IfAKA[6][1=\@empty, 3=\@empty]
1408 {%
1409    \protected@edef\arga{\trim@spaces{#1}}%
1410    \protected@edef\argc{\trim@spaces{#3}}%
1411    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1412    \def\csb{\@nameauth@Clean{#2}}%
1413    \def\csbc{\@nameauth@Clean{#2,#3}}%
1414    \def\csab{\@nameauth@Clean{#1!#2}}%
```

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

```
1415    \@nameauth@Error{#2}{macro \string\IfAKA}%
1416    \ifx\arga\@empty
1417      \ifx\argc\@empty
1418        \ifcsname\csb!PN\endcsname
1419          \edef\testex{\csname\csb!PN\endcsname}%
1420          \ifx\testex\@empty{#4}\else{#6}\fi
1421        \else{#5}\fi
1422      \else
1423        \ifx\suffb\@empty
1424          \ifcsname\csbc!PN\endcsname
1425            \edef\testex{\csname\csbc!PN\endcsname}%
1426            \ifx\testex\@empty{#4}\else{#6}\fi
1427          \else{#5}\fi
1428        \else
1429          \ifcsname\csb!PN\endcsname
1430            \edef\testex{\csname\csb!PN\endcsname}%
1431            \ifx\testex\@empty{#4}\else{#6}\fi
1432          \else{#5}\fi
1433        \fi
1434      \fi
1435    \else
1436      \ifcsname\csab!PN\endcsname
1437        \edef\testex{\csname\csab!PN\endcsname}%
1438        \ifx\testex\@empty{#4}\else{#6}\fi
1439      \else{#5}\fi
1440    \fi
1441 }
```

Changing Name Decisions

\ForgetName   This undefines a control sequence to force a "first use."

```
1442 \newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]
1443 {%
1444    \protected@edef\arga{\trim@spaces{#1}}%
1445    \protected@edef\argc{\trim@spaces{#3}}%
```

```
1446    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1447    \def\csb{\@nameauth@Clean{#2}}%
1448    \def\csbc{\@nameauth@Clean{#2,#3}}%
1449    \def\csab{\@nameauth@Clean{#1!#2}}%
1450    \@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`).

```
1451    \ifx\arga\@empty
1452      \ifx\argc\@empty
1453        \if@nameauth@LocalNames
1454          \if@nameauth@MainFormat
1455            \global\csundef{\csb!MN}%
1456          \else
1457            \global\csundef{\csb!NF}%
1458          \fi
1459        \else
1460          \global\csundef{\csb!MN}%
1461          \global\csundef{\csb!NF}%
1462        \fi
1463      \else
1464        \ifx\suffb\@empty
1465          \if@nameauth@LocalNames
1466            \if@nameauth@MainFormat
1467              \global\csundef{\csbc!MN}%
1468            \else
1469              \global\csundef{\csbc!NF}%
1470            \fi
1471          \else
1472            \global\csundef{\csbc!MN}%
1473            \global\csundef{\csbc!NF}%
1474          \fi
1475        \else
1476          \if@nameauth@LocalNames
1477            \if@nameauth@MainFormat
1478              \global\csundef{\csb!MN}%
1479            \else
1480              \global\csundef{\csb!NF}%
1481            \fi
1482          \else
1483            \global\csundef{\csb!MN}%
1484            \global\csundef{\csb!NF}%
1485          \fi
1486        \fi
1487      \fi
1488    \else
1489      \if@nameauth@LocalNames
1490        \if@nameauth@MainFormat
1491          \global\csundef{\csab!MN}%
1492        \else
1493          \global\csundef{\csab!NF}%
1494        \fi
1495      \else
1496        \global\csundef{\csab!MN}%
1497        \global\csundef{\csab!NF}%
1498      \fi
```

```
1499   \fi
1500 }
```

\SubvertName   This defines a control sequence to force a "subsequent use."

```
1501 \newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
1502 {%
1503   \protected@edef\arga{\trim@spaces{#1}}%
1504   \protected@edef\argc{\trim@spaces{#3}}%
1505   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1506   \def\csb{\@nameauth@Clean{#2}}%
1507   \def\csbc{\@nameauth@Clean{#2,#3}}%
1508   \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them.

```
1509   \@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.

```
1510   \ifx\arga\@empty
1511     \ifx\argc\@empty
1512       \if@nameauth@LocalNames
1513         \if@nameauth@MainFormat
1514           \csgdef{\csb!MN}{}%
1515         \else
1516           \csgdef{\csb!NF}{}%
1517         \fi
1518       \else
1519         \csgdef{\csb!MN}{}%
1520         \csgdef{\csb!NF}{}%
1521       \fi
1522     \else
1523       \ifx\suffb\@empty
1524         \if@nameauth@LocalNames
1525           \if@nameauth@MainFormat
1526             \csgdef{\csbc!MN}{}%
1527           \else
1528             \csgdef{\csbc!NF}{}%
1529           \fi
1530         \else
1531           \csgdef{\csbc!MN}{}%
1532           \csgdef{\csbc!NF}{}%
1533         \fi
1534       \else
1535         \if@nameauth@LocalNames
1536           \if@nameauth@MainFormat
1537             \csgdef{\csb!MN}{}%
1538           \else
1539             \csgdef{\csb!NF}{}%
1540           \fi
1541         \else
1542           \csgdef{\csb!MN}{}%
1543           \csgdef{\csb!NF}{}%
1544         \fi
1545       \fi
1546     \fi
1547   \else
```

```
1548      \if@nameauth@LocalNames
1549        \if@nameauth@MainFormat
1550          \csgdef{\csab!MN}{}%
1551        \else
1552          \csgdef{\csab!NF}{}%
1553        \fi
1554      \else
1555        \csgdef{\csab!MN}{}%
1556        \csgdef{\csab!NF}{}%
1557      \fi
1558    \fi
1559 }
```

Alternate Names

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

```
1560 \newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]
1561 {%
```

Prevent entering \AKA via itself or \@nameauth@Name. Prevent the index-only flag. Tell the
formatting system that \AKA is running.

```
1562    \if@nameauth@BigLock\@nameauth@Locktrue\fi
1563    \unless\if@nameauth@Lock
1564      \@nameauth@Locktrue%
1565      \@nameauth@InAKAtrue%
1566      \if@nameauth@OldReset
1567        \@nameauth@JustIndexfalse%
1568      \else
1569        \global\@nameauth@JustIndexfalse%
1570      \fi
```

Test for malformed input.

```
1571      \@nameauth@Error{#2}{macro \string\AKA}%
1572      \@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.

```
1573      \leavevmode\hbox{}%
1574      \protected@edef\argi{\trim@spaces{#1}}%
1575      \protected@edef\rooti{\@nameauth@Root{#2}}%
1576      \protected@edef\suffi{\@nameauth@Suffix{#2}}%
1577      \@nameauth@Parse[#3]{#4}[#5]{!PN}%
```

Create an index cross-reference based on the arguments.

```
1578      \unless\if@nameauth@SkipIndex
1579        \ifx\argi\@empty
1580          \ifx\suffi\@empty
1581            \IndexRef[#3]{#4}[#5]{\rooti}%
1582          \else
1583            \IndexRef[#3]{#4}[#5]{\rooti\space\suffi}%
1584          \fi
1585        \else
1586          \ifx\suffi\@empty
1587            \IndexRef[#3]{#4}[#5]{\rooti,\space\argi}%
1588          \else
1589            \IndexRef[#3]{#4}[#5]{\rooti,\space\argi,\space\suffi}%
1590          \fi
```

```
1591        \fi
1592      \fi
```

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

```
1593      \if@nameauth@OldReset
1594        \@nameauth@SkipIndexfalse%
1595        \@nameauth@AltAKAfalse%
1596        \@nameauth@NBSPfalse%
1597        \@nameauth@NBSPXfalse%
1598        \@nameauth@DoCapsfalse%
1599        \@nameauth@Accentfalse%
1600        \@nameauth@AllThisfalse%
1601        \@nameauth@ShowCommafalse%
1602        \@nameauth@NoCommafalse%
1603        \@nameauth@RevThisfalse%
1604        \@nameauth@RevThisCommafalse%
1605        \@nameauth@ShortSNNfalse%
1606        \@nameauth@EastFNfalse%
1607      \else
1608        \global\@nameauth@SkipIndexfalse%
1609        \global\@nameauth@AltAKAfalse%
1610        \global\@nameauth@NBSPfalse%
1611        \global\@nameauth@NBSPXfalse%
1612        \global\@nameauth@DoCapsfalse%
1613        \global\@nameauth@Accentfalse%
1614        \global\@nameauth@AllThisfalse%
1615        \global\@nameauth@ShowCommafalse%
1616        \global\@nameauth@NoCommafalse%
1617        \global\@nameauth@RevThisfalse%
1618        \global\@nameauth@RevThisCommafalse%
1619        \global\@nameauth@ShortSNNfalse%
1620        \global\@nameauth@EastFNfalse%
1621      \fi
1622      \@nameauth@Lockfalse%
1623      \@nameauth@InAKAfalse%
```

Close the "locked" branch.

```
1624    \fi
```

Call the full stop detection.

```
1625    \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
1626 }
```

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

```
1627 \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.

```
1628 \newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]
1629 {%
1630    \@nameauth@JustIndexfalse%
1631    \if@nameauth@SkipIndex
1632      \NameauthName[#1]{#2}\space(\SkipIndex\AKA[#1]{#2}[#3]{#4}[#5])%
1633    \else
1634      \NameauthName[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
1635    \fi
1636 }
```

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

```
1637 \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 >.

```
1638 \newenvironment{nameauth}{%
1639   \begingroup%
1640   \let\ex\expandafter%
1641   \csdef{<}##1&##2&##3&##4>{%
1642     \protected@edef\@arga{\trim@spaces{##1}}%
1643     \protected@edef\@testb{\trim@spaces{##2}}%
1644     \protected@edef\@testd{\trim@spaces{##4}}%
1645     \@nameauth@etoksb\expandafter{##2}%
1646     \@nameauth@etoksc\expandafter{##3}%
1647     \@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.

```
1648     \ifx\@arga\@empty
1649       \PackageError{nameauth}%
1650       {environment nameauth: Control sequence missing}%
1651     \fi
1652     \@nameauth@Error{##3}{environment nameauth}%
1653     \ifcsname\@arga\endcsname
1654       \PackageWarning{nameauth}%
1655       {environment nameauth: Shorthand macro already exists}%
1656     \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.

```
1657     \ifx\@testd\@empty
1658       \ifx\@testb\@empty
1659       \ex\csgdef\ex{\ex\@arga\ex}%
1660         \ex{\ex\NameauthName\ex{\the\@nameauth@etoksc}}%
1661       \ex\csgdef\ex{\ex L\ex\@arga\ex}%
1662         \ex{\ex\@nameauth@FullNametrue%
1663         \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
1664       \ex\csgdef\ex{\ex S\ex\@arga\ex}%
1665         \ex{\ex\@nameauth@FirstNametrue%
1666         \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
1667       \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.

```
1668         \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1669         \ex\ex\ex{\ex\ex\ex\NameauthName%
1670         \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
1671         \ex{\the\@nameauth@etoksc}}%
1672         \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
1673         \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1674         \ex\ex\ex\NameauthLName%
1675         \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
1676         \ex{\the\@nameauth@etoksc}}%
1677         \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
```

113

```
1678            \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1679            \ex\ex\ex\NameauthFName%
1680            \ex\ex\ex[\ex\the\ex\@nameauth@etoksb\ex]%
1681            \ex{\the\@nameauth@etoksc}}%
1682        \fi
1683      \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.

```
1684        \ifx\@testb\@empty
1685          \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1686            \ex\ex\ex{\ex\ex\ex\NameauthName%
1687            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1688            \ex[\the\@nameauth@etoksd]}%
1689          \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\@arga%
1690            \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FullNametrue%
1691            \ex\ex\ex\NameauthLName%
1692            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1693            \ex[\the\@nameauth@etoksd]}%
1694          \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\@arga%
1695            \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\@nameauth@FirstNametrue%
1696            \ex\ex\ex\NameauthFName%
1697            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1698            \ex[\the\@nameauth@etoksd]}%
1699        \else
```

Here are Western names with alternates. We have three arguments to expand, so we have seven, three, and one respective use of `\expandafter`.

```
1700          \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1701            \ex\ex\ex\ex\ex\ex\ex\@arga\ex\ex\ex\ex\ex\ex\ex}%
1702            \ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\NameauthName%
1703            \ex\ex\ex\ex\ex\ex[\ex\ex\ex\the%
1704            \ex\ex\ex\@nameauth@etoksb\ex\ex\ex]%
1705            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1706            \ex[\the\@nameauth@etoksd]}%
1707          \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1708            \ex\ex\ex\ex\ex\ex\ex L\ex\ex\ex\ex\ex\ex\ex\@arga%
1709            \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
1710            \ex\ex\ex\ex\ex\ex\@nameauth@FullNametrue%
1711            \ex\ex\ex\ex\ex\ex\NameauthLName%
1712            \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
1713            \@nameauth@etoksb\ex\ex\ex]%
1714            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1715            \ex[\the\@nameauth@etoksd]}%
1716          \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
1717            \ex\ex\ex\ex\ex\ex\ex S\ex\ex\ex\ex\ex\ex\ex\@arga%
1718            \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
1719            \ex\ex\ex\ex\ex\ex\@nameauth@FirstNametrue%
1720            \ex\ex\ex\ex\ex\ex\ex\NameauthFName%
1721            \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex%
1722            \@nameauth@etoksb\ex\ex\ex]%
1723            \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}%
1724            \ex[\the\@nameauth@etoksd]}%
1725        \fi
1726      \fi\ignorespaces%
1727    }\ignorespaces%
1728 }{\endgroup\ignorespaces}
```

4 Change History

5 Index

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