

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

Charles P. Schaum[†]

Released 2015/11/29

Abstract

The `nameauth` package automates the formatting and indexing of names. This aids the use of a **name authority** and the process of textual reordering and revision without needing to retype name references.

Contents

1	Introduction	2	2.5.9 Custom Formatting	20
1.1	Preliminaries	2	2.5.10 Disable Formatting	21
1.2	What's In A Name?	3	2.6 Tweaks: <code>\ForgetName</code> and <code>\SubvertName</code>	22
2	Usage	4	2.7 Name Variant Macros	22
2.1	Package Options	4	2.7.1 <code>\AKA</code>	22
2.2	Quick Start Guide	6	2.7.2 <code>\PName</code>	24
2.2.1	Main Interface	6	2.8 Indexing Macros	25
2.2.2	Simplified Interface	7	2.8.1 <code>\IndexName</code>	25
2.2.3	Older Syntax	9	2.8.2 <code>\TagName</code>	25
2.2.4	Tips and Warnings	10	2.8.3 <code>\UntagName</code>	26
2.3	Naming Macros	12	2.8.4 Global Name Exclusion	26
2.3.1	<code>\Name</code> and <code>\Name*</code>	12	2.8.5 Indexing Control	27
2.3.2	Forenames: <code>\FName</code>	13	2.9 Variant Spellings	27
2.4	Affixes and Eastern Names	14	2.10 Naming Pattern Reference	28
2.4.1	Affixes Need Commas	14	2.10.1 Basic Naming	28
2.4.2	Eastern Names	15	2.10.2 Particles	31
2.5	Other Naming Topics	16	3 Implementation	32
2.5.1	Fault Tolerance	16	3.1 Boolean Values	32
2.5.2	Listing by Surname	16	3.2 Package Options	33
2.5.3	Naming Standards	16	3.3 Internal Macros	34
2.5.4	Hyphenation	17	3.4 User Interface Macros	37
2.5.5	Indexing and <code>babel</code>	17	4 Change History	63
2.5.6	Detecting Punctuation	17	5 Index	65
2.5.7	Accented Names	18		
2.5.8	Index Sorting	19		

This manual is designed with a text block compatible with both US letter and A4. Macro cross refs are minimized to give a relatively “clean” index that illustrates this package’s indexing functions.

*This file describes version v2.11, last revised 2015/11/29.

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

1 Introduction

1.1 Preliminaries

Books can reference hundreds of names. It takes time and money to check them. This package helps to format and index names consistently and automatically, helping you save time and money. Features include:

- Simultaneously format, display, and index names using fewest keystrokes.
- Context automatically determines the syntactic form of names.
- The default uses longer name forms for the first use and shorter forms thereafter.
- Apply typesetting formats to fit your needs without retyping names.
- Move text in the document without retyping names.
- Show name variants in the text, yet index consistent name forms.
- Handle some cross-cultural naming conventions.
- Allow for different capitalizing and other conventions.
- Index different people with the same name by using tags.
- Process names in list environments and other special environments.

I started using \LaTeX and wrote this package when translating old German and Latin texts. I learned much more than I expected regarding \LaTeX typesetting engines, font systems, indexing programs, class and package interactions, and naming conventions. Please consider the following general notes that apply to the development and use of this package:

As of version 2.0 you *can* put control sequences in names, but with a few caveats; see Section 2.5.7. Also, this package is more fault-tolerant and prevents or warns about malformed input.

This manual performs something of a “torture test” on this package. You might want to avoid doing that.

This package depends on `etoolbox`, `ifxetex`, `ifluatex`, `suffix`, `trimspaces`, and `xargs`. It has been tested with `latex`, `lualatex`, `pdflatex`, and `xelatex`. It will work with `makeindex` and `texindy`. This documentation was typeset with `pdflatex` and `makeindex`.

Indexing generally conforms to the standard in Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). This should be suitable for a very wide application across a number of disciplines.

Thanks to MARC VAN DONGEN, ENRICO GREGORIO, PHILIPP STEPHANI, HEIKO OBERDIEK, UWE LUECK, and ROBERT SCHLICHT for their assistance in the early versions of this package.

This documentation uses names of living and historical figures because users refer to real people in their projects. At no time do I intending any statement of bias for or against a particular person, culture, or tradition. All names mentioned herein deserve respect for the impact and legacy of their bearers.

1.2 What's In A Name?

Personal name forms are ambiguous apart from historical and cultural contexts. The `nameauth` package helps you encode names in a way that anticipates and responds automatically to such contexts without the need to retype names.

Below we see three categories of names with suggestions regarding what might be used therein.¹ Professional writing often calls for the full form of a person's name to be used in its first occurrence, with shorter forms used thereafter. This package adapts that principle to all the forms below.

1. Western name:

Forename(s)	Surname(s)	Sobriquet, etc.
/		\
Personal name(s):	Family designator:	Sobriquet / title:
<i>baptismal name</i>	<i>father's family</i>	<i>senior, junior, III...</i>
<i>Christian name</i>	<i>mother's family</i>	<i>notable feature</i>
<i>first and middle</i>	<i>ancestor</i>	<i>notable attribute</i>
<i>names</i>	<i>occupation</i>	<i>place of origin</i>
<i>praenomen</i>	<i>place of origin</i>	<i>territory</i>
	<i>territory</i>	<i>territory</i>
	<i>nomen/cognomen</i>	<i>agnomen</i>
	<i>patronym</i>	

2. Eastern name:

Family name	Given name
/	\
Family designator	(Multiple names are rare, but multi-character names do exist.)

3. Ancient name:

Given name	Sobriquet, etc.
/	\
Personal name	Sobriquet / title:
	<i>senior, junior, III...</i>
	<i>notable feature</i>
	<i>notable attribute</i>
	<i>place of origin</i>
	<i>territory</i>

¹These suggestions are not exhaustive. For special cases, one might have to decide how to handle, for example, Hungarian and Icelandic names as Eastern or Western.

2 Usage

2.1 Package Options

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

Package options address the following:

1. The way name data are entered
2. Semantic display of names (full/short, commas/none, reversed/caps)
3. Typographic display of names (formatted or not)
4. Indexing and sorting of names in the index

Default options are in boldface.

Show/Hide Affix Commas

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

This option is set at load time. If you use *modern standards* or Eastern names, 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:

- First, the **comma** option produces, *e.g.*, JAMES EARL CARTER, JR. Yet it limits the use of macros like `\AKA` and `\PName`.²
- Second, Section 2.4.1 shows how one can use `\ShowComma` with the **nocomma** option to get similar results, but with more flexibility.

Enable/Disable Formatting

mainmatter	Enable typographic formatting attributes (see formatting options below), starting at the beginning of a document.
frontmatter	Disable typographic formatting, but retain automatic full and short forms.

The default starts formatting names immediately. Use the **frontmatter** option to suppress name formatting until you want it to start via `\NamesActive`. These options have no additional effects on the index. See Section 2.5.10.

Enable/Disable Indexing

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

The default starts indexing names right away. The **noindex** option disables the indexing of names until `\IndexActive` enables it. This applies only to naming and indexing macros in the **nameauth** package. See Section 2.8.5.

²Before version 0.9 the **nameauth** package assumed the **comma** option.

Enable/Disable Index Sorting

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

The default allows `\PretagName` to create sort keys used in `makeindex` / `texindy`. Seldom would one change this option. See Section 2.5.8.

Capitalize Entire Surnames

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

This only affects names printed in the body text. To get caps in both the body text and the index, the user should type in the caps manually — an easy task with the simplified interface. Section 2.4.2 details macros that enable capitalization on a section-level and per-name basis.

Reverse Name Order

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

See Sections 2.4.1, 2.4.2, and 2.5.2 for related macros to control name reversing by section or by name. So-called “last-comma-first” lists of names are *not* the same as the `comma` option.

Formatting Attributes

alwaysformat	If formatting is enabled by the <code>mainmatter</code> option or by <code>\NamesActive</code> , this option causes all names to have special typographic formatting.
smallcaps	Set the first use of a name in small caps.
italic	Italicize the first occurrence of a name.
boldface	Set the first use of a name in boldface.
noformat	Do not define a default format. Can be used with custom formatting.

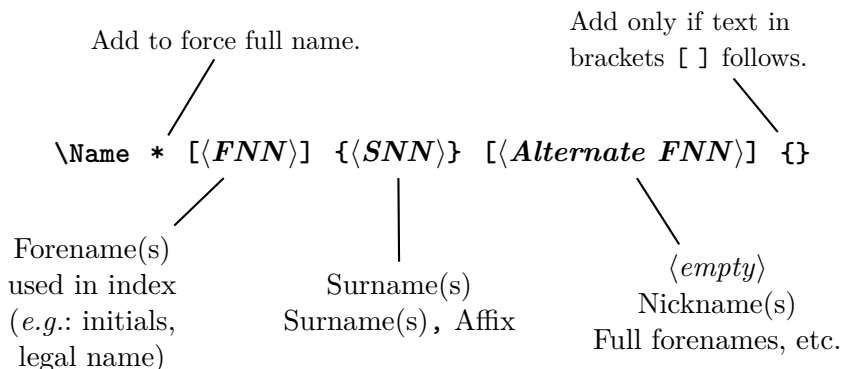
The default is `smallcaps` because this package was developed to aid the editing and translation of older German and Latin documents. Section 2.5.9 details even more possibilities for presenting the first use of names.

2.2 Quick Start Guide

2.2.1 Main Interface

See Section 2.3 for a proper description of `\Name`. Here we see briefly how to work with the classes of names in Section 1.2. We abbreviate the command parameters $\langle forename(s) \rangle$ with $\langle FNN \rangle$ and $\langle surname(s) \rangle$ with $\langle SNN \rangle$. The `nocomma` option works best with affixes, Eastern, and ancient names.

Western Names

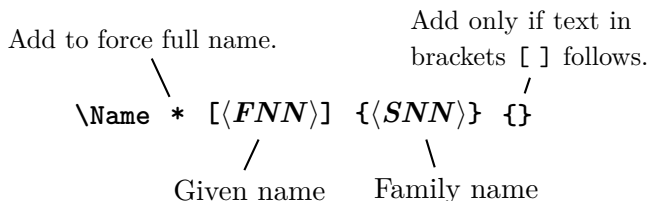


Usual forms:

```
\Name[ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ } \Name[George]{Washington}
\Name[ $\langle FNN \rangle$ ]{ $\langle SNN, Affix \rangle$ } \Name[John David]{Rockefeller, II}
You must include the  $\langle FNN \rangle$  field with alternate forenames. The  $\langle Alternate FNN \rangle$ 
are swapped with the  $\langle FNN \rangle$ , but only in the body text:
\Name[ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ }[ $\langle Alternate FNN \rangle$ ]
\Name[Bob]{Hope}[Leslie Townes]
\Name[ $\langle FNN \rangle$ ]{ $\langle SNN, Affix \rangle$ }[ $\langle Alternate FNN \rangle$ ]
\Name[John David]{Rockefeller, IV}[Jay]
```

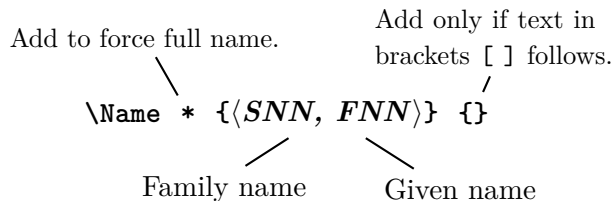
The older syntax is `\Name{ $\langle SNN \rangle$ }[$\langle Affix \rangle$]`. See Section 2.2.3 for its usage and its shortcomings. It remains for backward compatibility.

Eastern Names in the Text, Western-style Index



Technically, these are Western name forms without affixes. The reversing macros (Section 2.4.2) cause them to display in Eastern order in the body text only. The index entries are Western in fashion: $\langle SNN \rangle$, $\langle FNN \rangle$. This “non-native” form of Eastern names excludes all comma-delimited forms and the old syntax.

Eastern Names in the Text, Eastern-style Index



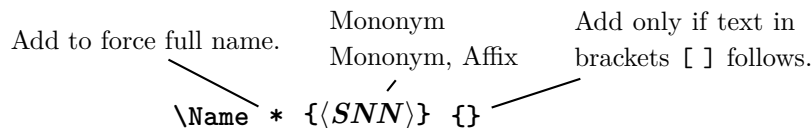
Usual form:

`\Name{\langle SNN, FNN \rangle}` `\Name{Yamamoto, Isoroku}`

These names truly are Eastern names. They will take the form $\langle SNN\ FNN \rangle$ in the index even if the reversing macros (Section 2.4.2) put the names in Western order in the body text. We call this the “native” form of Eastern names.

The old form of Eastern names is `\Name{\langle SNN \rangle}[\langle FNN \rangle]`. Again, this is retained only for backward compatibility. Cf. Section 2.2.3.

Ancient Names



Usual form:

`\Name{\langle SNN \rangle}` `\Name{Aristotle}`
`\Name{\langle SNN, affix \rangle}` `\Name{Elizabeth, I}`

These forms are used for royalty, ancient figures, and other mononyms.³ The older syntax is `\Name{\langle Mononym \rangle}[\langle Affix \rangle]`. Cf. Section 2.2.3.

2.2.2 Simplified Interface

nameauth The **nameauth** environment allows one to save typing and aid consistency by using shorthands. It produces control sequences that are fully compatible with the main interface. Although not required, **nameauth** is best used in the document preamble to avoid undefined control sequences.⁴

```
\begin{nameauth}
  \< \langle cseq1 \rangle & \langle arg1 \rangle & \langle arg2 \rangle & \langle arg3 \rangle >
  \< \langle cseq2 \rangle & \langle arg1 \rangle & \langle arg2 \rangle & \langle arg3 \rangle > ...
\end{nameauth}
```

Please bear in mind that `\<` in this context is a control sequence. If you forget it or just use `<` without the backslash you will get errors. Leading and trailing spaces in each `&`-delimited field are stripped, as is also the case in the main interface. There *must* be four argument fields (three ampersands) per line.

³Technically, the native Eastern forms and the $\langle Mononym, Affix \rangle$ forms are identical, although used in different contexts. You would not wish to reverse a royal name, for example.

⁴The **nameauth** environment uses `\ignorespaces` to mitigate the need for trailing `%`.

Each `\cseq` creates three macros. In the document text, `\cseq` itself calls `\Name`. `\Lcseq` (think “Long”) calls `\Name*`. `\Scseq` (think “Short”) calls `\FName`. In the document text, as with `\Name`, include trailing braces `{ }` if text in brackets `[]` follows any of the shorthands.

The following example illustrates a fairly complete set of names:

```
\begin{nameauth}
  \< Wash & George & Washington & >           Western
  \< Soto & Hernando & de Soto & >             particle
  \< JRock & John David & Rockefeller, II & >    affix
  \< JayR & John David & Rockefeller, IV & Jay >  nickname5
  \< Aris & & Aristotle & >                   ancient
  \< Eliz & & Elizabeth, I & >                 royal
  \< Atil & & Atilla, the Hun & >               ancient
  \< Konoe & Fumimaro & Konoe & >             Eastern/Western index
  \< Yamt & & Yamamoto, Isoroku & >           Eastern/Eastern index
\end{nameauth}
```

The following control sequences in the body text illustrate how this environment works. Please pay attention to first and subsequent uses, as well as how the L and S versions and other control sequences modify names:

Basic Uses:		Medieval/Royal:	
<code>\Wash</code>	GEORGE WASHINGTON	<code>\Eliz</code>	ELIZABETH I
<code>\LWash</code>	George Washington	<code>\Eliz</code>	Elizabeth
<code>\Wash</code>	Washington	<code>\Atil</code>	ATILLA THE HUN
<code>\SWash</code>	George	<code>\Atil</code>	Atilla
Western Reversed with Comma:		Western / Western Index:	
<code>\RevComma\LWash</code>	Washington, George	<code>\Konoe</code>	FUMIMARO KONOE
Particles:		<code>\LKonoe</code>	Fumimaro Konoe
<code>\Soto</code>	HERNANDO DE SOTO	<code>\Konoe</code>	Konoe
<code>\Soto</code>	de Soto	Eastern / Western Index:	
<code>\CapThis\Soto</code>	De Soto	<code>\CapName\RevName\LKonoe</code>	†KONOE Fumimaro
Affixes:		<code>\CapName\Konoe</code>	†KONOE
<code>\JRock</code>	JOHN DAVID ROCKEFELLER II	Eastern / Eastern Index:	
<code>\LJRock</code>	John David Rockefeller II	<code>\CapName\Yamt</code>	YAMAMOTO ISOROKU
<code>\JRock</code>	Rockefeller	<code>\CapName\LYamt</code>	YAMAMOTO Isoroku
Nicknames: (See Section 2.3.2)		<code>\CapName\Yamt</code>	YAMAMOTO
<code>\JayR</code>	JAY ROCKEFELLER IV	Western / Eastern Index:	
<code>\SJayR</code> <code>\JayR</code>	Jay Rockefeller	<code>\RevName\LYamt</code>	Isoroku Yamamoto
Ancient:		<code>\Yamt</code>	Yamamoto
<code>\Aris</code>	ARISTOTLE		
<code>\Aris</code>	Aristotle		

In this manual we use a dagger (†) to indicate all Eastern names with Western forms in the index. The user is encouraged to compare the index entries with the body text in order to understand better what the macros are doing.

⁵`\SJayR` always gives “Jay.” See Section 2.3.2 for possible difficulties when using this variation. Use `\AKA` to create a *see* reference from Jay Rockefeller to his full index entry: “Rockefeller, John David, IV.” For more on `\AKA` see Section 2.7.1.

2.2.3 Older Syntax

An “obsolete” syntax remains for backward compatibility with early versions of `nameauth` and with the `comma` option. Please avoid mixing the older and newer forms. The old form lacks some error checking and robustness contained in the new syntax and limits the use of several macros:

<code>\Name{Dagobert}[I]</code>	<i>royal name</i>
<code>\Name{Yoshida}[Shigeru]</code>	<i>Eastern name</i>
<code>\begin{nameauth}</code>	
<code>\< Dagb & & Dagobert & I ></code>	<i>royal name</i>
<code>\< Yosh & & Yoshida & Shigeru ></code>	<i>Eastern name</i>
<code>\end{nameauth}</code>	

`\Dagb` gives DAGOBERT I, while the next `\Dagb` produces Dagobert. In similar fashion as before, we see `\LDagb` Dagobert I, `\CapName\Yosh` YOSHIDA SHIGERU, and `\CapName\RevName\LYosh` Shigeru YOSHIDA.

- In the old syntax, `\Name{Henry}[VIII]` prints “HENRY VIII” and “Henry.” If you mix `\Name{Henry}[VIII]` with the newer `\Name{Henry, VIII}` they both print HENRY VIII and Henry in the body text. Yet they generate different control sequences for both first/subsequent uses and index tags.⁶
- Erroneously typing `\Name[Henry]{VIII}` prints “HENRY VIII” and “VIII,” as well as producing a malformed index entry. As of version 2.0, this kind of malformed input creates no side effects for well-formed input.
- Avoid forms like `\Name[Henry]{VIII}[Tudor]` which gives “Tudor VIII” and “VIII.” Again, this classifies as malformed input.
- Also avoid `\Name{Henry, VIII}[Tudor]` unless you really want “HENRY VIII TUDOR” and “Henry” in the body text and “Henry VIII” in the index.
- One solution to incorporate the house designation adds “Tudor” as needed in the text to `\Name{Henry, VIII}` and uses a tag in the index: `\TagName{Henry, VIII}{ Tudor}` (see Section 2.8.2).
- The older syntax will not work with some macros. From the film *Men in Black III*, `\AKA{Boris}[the Animal]{Just Boris}` fails. `\PName` fails for the same reasons. See also Section 2.7.1
- This form does work: `\Name{Boris, the Animal} \AKA{Boris, the Animal}{Just Boris}`. You get BORIS THE ANIMAL being “Just Boris.”

⁶Technically you can mix the two, as I do here. This package is all about offering choices, not restricting authors. Yet you must force first and subsequent uses with `\ForgetName` and `\SubvertName`, and make common index tags, *e.g.*: `\TagName{Henry, VIII}{, king}` and `\TagName{Henry}[VIII]{, king}`. That undermines the time-saving features offered by this package. Since authors have personal styles and workflows, this package tries its best to embrace the horror of inherent ambiguity in names and their usage.

2.2.4 Tips and Warnings

- Keep it simple! Avoid unneeded macros and use the simplified interface.
- Check braces and brackets with naming macros to avoid errors like “Paragraph ended...” and “Missing *⟨grouping token⟩* inserted.”
- For stage names, etc., try using *e.g.*: `\Name[J.]{Kreskin}[The Amazing] (\AKA[J.]{Kreskin}[Joseph]{Kresge})`. As a result you get THE AMAZING KRESKIN (Joseph Kresge). The corresponding index entry will be “Kreskin, J.” You must have a name in the first optional field for this to work.
- Let this package help your workflow as you need it, instead of trying to figure out all its intricacies. Special cases like “Iron Mike” Tyson as the nickname for MIKE TYSON may be handled in a number of ways.
 1. Follow ‘‘Iron Mike’’ with `\IndexName[Mike]{Tyson}` and do whatever you want in the text.
 2. `\SubvertName[Mike]{Tyson}\FName[Mike]{Tyson}[Iron Mike]\Name[Mike]{Tyson}` gives “Iron Mike Tyson.” You are responsible for typesetting the first use and you can `\let` the whole thing to a control sequence.
 3. Use ‘‘`\AKA[Mike]{Tyson}{Iron Mike}`’’ to create “Iron Mike” in the text and a *see*-type cross-reference to “Tyson, Mike” in the index. Be sure to have an occurrence of `\Name[Mike]{Tyson}` in the text. See also Section 2.7.1.
- Omit spaces between initials if possible; see also Bringhurst’s *Elements of Typographic Style*. If your publisher wants spaces between initials, use `\frenchspacing`⁷ or put thin spaces `\,` between them. Consider:

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

Compare the following:

<code>\White</code> and <code>\LWhite</code>	(E. B. WHITE and E. B. White)
Normal:	E. B. WHITE and E. B. White.

- One way to spot errors is to compare index entries with names in the body text. All macros that produce output also emit meaningful warnings. `\PName` produces warnings via `\Name` and `\AKA`.
- Please pay greater attention to the warnings produced by `\IndexName`, `\TagName`, `\UntagName`, and `\ExcludeName`. Many other warnings are FYI.
- The `nameauth` environment only generates a warning when you overwrite a control sequence designator with a new name.
- Inserting an empty string in a required field will generate an error.

⁷Although there should be no difference between `\frenchspacing` and `\nonfrenchspacing` when it comes to initials, some classes or packages may affect that.

Warnings result from:

- Using a cross-reference [*alternate FNN*]{*alternate SNN*}[*alt. names*] created by \AKA as a name reference in \Name, \FName, and \PName. They merely will print a name in the body text.
- Using a name reference [*FNN*]{*SNN*}[*Alternate names*] created by \Name, \FName, and \PName as a cross-reference in \AKA. It merely will print a name in the body text.
- Using \AKA to create the same cross-reference multiple times or with a cross-reference created by \ExcludeName. It merely will print a name in the body text, but not the index.
- Using \IndexName to index a cross-reference made via \AKA or via the mechanism in \ExcludeName as a main entry. It will do nothing.
- Using \TagName, \UntagName, and \PretagName with cross-references. The first two will do nothing. However, \PretagName will “pretag” a cross-reference. This is the desired behavior.
- Using \ExcludeName with cross-references. It will do nothing.
- Using \ExcludeName to exclude a name that has already been used. Likewise, it will do nothing.
- Using \Name, \FName, \PName, and \AKA to refer to names and cross-references excluded by \ExcludeName. They merely will print a name in the body text.
- Using the nameauth environment to redefine shorthands, such as:

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

2.3 Naming Macros

2.3.1 `\Name` and `\Name*`

`\Name` This macro generates two forms of the name: a printed form in the text and a
`\Name*` form of the name that occurs in the index. The general syntax is:

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

Here we see how the syntax works:

<code>\Name[Albert]{Einstein}</code>	ALBERT EINSTEIN
<code>\Name*[Albert]{Einstein}</code>	Albert Einstein
<code>\Name[Albert]{Einstein}</code>	Einstein
<code>\Name{Confucius}</code>	CONFUCIUS
<code>\Name*{Confucius}</code>	Confucius
<code>\Name{Confucius}</code>	Confucius
<code>\Name[M.T.]{Cicero}[Marcus Tullius]</code>	MARCUS TULLIUS CICERO
<code>\Name*[M.T.]{Cicero}[Marcus Tullius]</code>	Marcus Tullius Cicero
<code>\Name[M.T.]{Cicero}[Marcus Tullius]</code>	Cicero
<code>\Name{Charles, the Bald}</code>	CHARLES THE BALD
<code>\Name*{Charles, the Bald}</code>	Charles the Bald
<code>\Name{Charles, the Bald}</code>	Charles

`\Name` displays and indexes names, as illustrated in Section 2.10. It always prints the `⟨SNN⟩` field. `\Name` prints the “full name” at the first occurrence, then the partial form thereafter. `\Name*` always prints the full name.

The `⟨Alternate names⟩` field replaces the `⟨FNN⟩` field in the text only if the `⟨FNN⟩` field is not empty; see “Cicero” above. One can use a nickname in some instances while keeping the indexed form constant.

Thus, regarding their index entries and first/subsequent uses in `nameauth`, `\Name[M.T.]{Cicero}[Marcus Tullius]` and `\Name[M.T.]{Cicero}` are equivalent, while `\Name{Cicero}[Marcus Tullius]` and `\Name{Cicero}` are not. Generally avoid older forms like `\Name{Charles}[the Bald]`.

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

Above we see the same setup with the simplified interface. In the body text, `\Einstein`, `\LEinstein`, and `\Einstein` produce ALBERT EINSTEIN, Albert Einstein, and Einstein. `\CBald` and `\CBald` give CHARLES THE BALD and Charles. The next section demonstrates that `\LCicero[Marcus Tullius]` allows for M.T. CICERO to be both Marcus Tullius Cicero and M.T. Cicero.

2.3.2 Forenames: \FName

`\FName` This casual friend of `\Name` prints only “first” names, but it will still print a full name when a first use occurs. `\FName` is intended for Western-style names. `\FName*` is only a synonym for `\FName`. The syntax is basically the same:

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

The first reference to `\FName` always is a full name. That prevents a first-name-only reference before a person has been introduced. Intentionally, `\FName` *never* gives the first name with Eastern names. For examples we see below:

<code>\FName[Albert]{Einstein}</code>	ALBERT EINSTEIN
<code>\FName[Albert]{Einstein}</code>	Albert
<code>\FName{Confucius}</code>	CONFUCIUS
<code>\FName{Confucius}</code>	Confucius
<code>\FName[M.T.]{Cicero}[Marcus Tullius]</code>	MARCUS TULLIUS CICERO
<code>\FName[M.T.]{Cicero}[Marcus Tullius]</code>	Marcus Tullius
<code>\FName{Charles, the Bald}</code>	CHARLES THE BALD
<code>\FName{Charles, the Bald}</code>	Charles

The Cicero example shows how the `⟨Alternate names⟩` field can work. Be careful with nicknames: ‘‘`\FName[Chesley B.]{Sullenberger, III}[Sully]`’’ produces “SULLY SULLENBERGER III” and “Sully.” This is not a “bug” as such, as the Cicero example illustrates. Names are very context-sensitive.

With `\SEinstein`, `\SConfucius`, `\SCicero`, and `\SCBald` we get Albert, Confucius, M.T., and Charles when using the simplified interface example from the previous page. `\SCicero[Marcus Tullius]` gives Marcus Tullius. The simplified interface allows you to use a “default nickname.” In Section 2.2.2 `\SJAYR` gives JAY ROCKEFELLER IV and Jay, but the index entry “Rockefeller, John David, IV.” Yet default nicknames are not always what they seem:

```
\begin{nameauth}
  < Ches & Chesley B. & Sullenberger, III & >
  < Sully & Chesley B. & Sullenberger, III & Sully >
\end{nameauth}
```

The first use `\Ches` prints “CHESLEY B. SULLENBERGER III.” Later, `\SChes` and `\SSully` print “Chesley B.” and “Sully.” While `\SChes[Sully]` always gives, “Sully,” `\SSully[Chesley B.]` prints “Sully[Chesley B.]”

`\SSully[Chesley B.]` expands to what it should be, not what you might expect it to be: `\FName[Chesley B.]{Sullenberger, III}[Sully][Chesley B.]`. Thus we prefer `\LCicero[Marcus Tullius]` and `\SCicero[Marcus Tullius]`:

```
\begin{nameauth}
  < Cicero & M.T. & Cicero & > preferred
  < Cicero & M.T. & Cicero & Marcus Tullius > not preferred
\end{nameauth}
```

This package offers preferred choices, but its design does not force you to use them because names and their uses have many variants.

2.4 Affixes and Eastern Names

2.4.1 Affixes Need Commas

Comma-delimited affixes handle several different name types. *Always include a comma as an affix delimiter*, even when the `nocomma` option does not print the comma. Extra spaces and trailing commas are ignored. Other name types include royal, medieval, and Eastern names:

<code>\Name[Oskar]{Hammerstein, II}</code>	OSKAR HAMMERSTEIN II
<code>\Name[Oskar]{Hammerstein, II}</code>	Hammerstein
<code>\Name{Louis, XIV}</code>	LOUIS XIV
<code>\Name{Louis, XIV}</code>	Louis
<code>\Name{Sun, Yat-sen}</code>	SUN YAT-SEN
<code>\Name{Sun, Yat-sen}</code>	Sun

One must use comma-delimited suffixes when cross-referencing affixed Western names, royal names, some medieval names, and Eastern names with `\AKA`; see Section 2.7.1.

`\KeepAffix` Put `\KeepAffix` before `\Name` or `\AKA` if a line break or page break divides a $\langle SNN, affix \rangle$ pair. This puts a non-breaking space between *SNN* and *affix* in the body text, but not in the index. Other options to fix bad breaks include using `\hbox`, kerning and spacing in the `microtype` package, etc.

`\ShowComma` The `comma` option is restrictive and used to reproduce older texts. `\ShowComma` gets the same results on a per-name basis while using the default `nocomma` option. With `\ShowComma\Name[Louis]{Gossett, Jr.}` one gets LOUIS GOSSETT, JR. One must use `\ShowComma` consistently or risk errors in the body text and index.

Avoid using the older syntax, shown below, except with the `comma` option. It does not handle Western names with affixes and some other name types. `\AKA` and `\PName` cannot create cross-references to these forms. These older forms include:

<code>\Name{Henry}[VIII]</code>	HENRY VIII
<code>\Name{Henry}[VIII]</code>	Henry
<code>\Name{Chiang}[Kai-shek]</code>	CHIANG KAI-SHEK
<code>\Name{Chiang}[Kai-shek]</code>	Chiang

These older forms work because no $\langle FNN \rangle$ are present. Otherwise you would get weird nicknames. Again, please avoid using the older syntax.

2.4.2 Eastern Names

The `nameauth` package offers “non-native” and “native” ways to handle romanized Eastern names. `\RevName\Name[⟨Eastern FNN⟩]{⟨Eastern SNN⟩}` will produce an Eastern name in the body text and the Western form $\langle SNN \rangle$, $\langle FNN \rangle$ in the index, including the comma. We call this “non-native” mode.

In contrast, both `\Name{⟨Eastern SNN, Eastern FNN⟩}` and the older syntax `\Name{⟨Eastern SNN⟩}[⟨Eastern FNN⟩]` produce an Eastern name form in the body text: $\langle SNN \rangle \langle FNN \rangle$ as well as in the index. This form has no comma in the index. We call this “native” mode. Offering these two modes gives the greatest flexibility in indexing requirements.

`\ReverseActive` The “smart” reverse output mechanism converts between Western and Eastern forms in the text, but not the index. If one wants a Western-format index, then
`\ReverseInactive` pick non-native mode. If Eastern forms are okay in the index, then pick native
`\RevName` mode. In addition to the class options described in Section 2.1, `\ReverseActive`
and `\ReverseInactive` toggle reversing on a larger scale, while `\RevName` is used
once per `\Name`.

This list of Japanese music artists shows `\RevName` in action. Names in Western order, then non-native Eastern order are marked with a dagger (†). All other names are in native Eastern, then Western order. Forms using the old syntax are in parentheses. Name formatting is turned off in order to focus on reversing:

	<i>unchanged</i>	<code>\RevName</code>
<code>†\Name*[Aiko]{Nakano}</code>	†Aiko Nakano	†Nakano Aiko
<code>\Name*{Arai, Akino}</code>	Arai Akino	Akino Arai
<code>(\Name*{Ishida}[Yoko])</code>	(Ishida Yoko)	(Yoko Ishida)
<code>\Name*{Yohko}</code>	Yohko	Yohko

`\AllCapsActive` Use `\AllCapsActive`, `\AllCapsInactive`, and `\CapName` for fully-capitalized
`\AllCapsInactive` family names in the body text. These macros are analogous to the reversing
`\CapName` macros above and may be used alone or with those and other state-toggling
macros, *e.g.* `\CapName\RevName\Name`. Names in Western order, then non-native
Eastern order are marked with a dagger (†). All other names are in native Eastern,
then Western order. Forms using the old syntax are in parentheses. Name
formatting is turned off in order to focus on capitalizing and reversing:

	<i>unchanged</i>	<code>\CapName\RevName</code>
<code>†\Name*[Yoko]{Kanno}</code>	†Yoko KANNO	†KANNO Yoko
<code>\Name*{Shikata, Akiko}</code>	SHIKATA Akiko	Akiko SHIKATA
<code>(\Name*{Nogawa}[Sakura])</code>	(NOGAWA Sakura)	(Sakura NOGAWA)
<code>\Name*{Yohko}</code>	YOHKO	YOHKO

Notice how capitalization is independent of formatting. The reversing and capitalization macros also work with `\AKA`. They affect only the text, not the index. Whoever wants all-cap forms in the index will have to capitalize everything manually or modify the macros.

2.5 Other Naming Topics

2.5.1 Fault Tolerance

Especially since version 2.0, the `nameauth` package tries to prevent malformed input from creating side effects. For example, the malformed `\Name[Henry]{VIII}` and the well-formed `\Name{Henry}[VIII]` used to create the same control sequence and thus affect each other. That no longer happens. Furthermore, we guard against empty required values being passed to naming macros.

To reduce errors, `\Name`, `\FName`, `\AKA`, and `\IndexName` ignore leading and trailing spaces — but not medial spaces — making the following equivalent:

Macro Example	Resulting text
<code>\Name*[Martin Luther]{King, Jr.}</code>	MARTIN LUTHER KING JR.
<code>\Name*[Martin Luther]{King, Jr.}</code>	Martin Luther King Jr.
<code>\Name*[Martin Luther]{King, Jr.}</code>	Martin Luther King Jr.
<code>\Name*[Martin Luther]{ King, Jr.}</code>	Martin Luther King Jr.
<code>\Name*[Martin Luther]{King, Jr. }</code>	Martin Luther King Jr.
<code>\Name*[Martin Luther]{ King, Jr. }</code>	Martin Luther King Jr.

2.5.2 Listing by Surname

`\ReverseCommaActive` The reversing macros `\ReverseCommaActive`, `\ReverseCommaInactive`, and `\ReverseCommaInactive` `\RevComma` allow the easy generation of name lists ordered as $\langle surname \rangle$, $\langle forename(s) \rangle$. The first two are broad toggles, while the third works on a per-name basis. Eastern, medieval, and royal names do not work with these macros. Name formatting has been turned off to focus on reversing and commas:

John Stuart Mill	Mill, John Stuart	OK
Oskar Hammerstein II	Hammerstein II, Oskar	OK
John Eriugena	Eriugena John	incompatible
Mao Tse-tung	Tse-tung Mao	incompatible
Anaximander	Anaximander	OK

2.5.3 Naming Standards

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

`\CapThis` In English, these particles go in the $\langle SNN \rangle$ field of `\Name`, e.g., WALTER DE LA MARE. To capitalize *de* when it arises at the beginning of a sentence, use `\CapThis\Name[Walter]{de la Mare}`. De la Mare will think it fair. In the rare case where the particle is but one character (unlike all those listed above), the capping macros will eat the space between particle and name. One workaround is to follow such a particle with `\nobreakspace`.

`\AccentCapThis` `\CapThis` works in all cases under `xelatex` and `lualatex`. Using `pdflatex` causes `\CapThis` to fail and halt execution only when the initial character in $\langle SNN \rangle$ is an extended Unicode character; see Section 2.5.7. In this special and infrequent case, either substitute `\AccentCapThis`, or put the extended character in braces, or use control sequences.

Non-English contexts do not always bind particles to surnames. Using `\Name` and `\FName` with alternate forenames helps address this. See also Section 2.10.2.

2.5.4 Hyphenation

The simplified interface trivializes the consistent insertion of optional hyphens in names, as we see below:

```
\begin{nameauth}
  < Bier & Johann & Bier\~mann & >
\end{nameauth}
```

We get JOHANN BIERMANN and Biermann. This should prevent the break “Biermann,” which could happen otherwise. You can even tag and untag such forms. The bad break above was manufactured, while the bad break below is actual.

Bad breaks can be fixed with the `babel` or `polyglossia` packages. JOHN STRIETELMEIER can have a bad break in English, as you see. Using `babel`, we can use the following example so that `\de{\Name*[John]{Strietelmeier}}` generates John Strietelmeier and helps prevent bad breaks:

```
\newcommand{\de}[1]{\foreignlanguage{ngerman}{#1}}
```

2.5.5 Indexing and `babel`

`texindy` Using `babel` with Roman page numbers will put `\textlatin` in the index entries if one includes a language that does not use the Latin alphabet—even if the main language does. The `texindy` program will ignore such references. This issue can affect `nameauth`. One workaround for `texindy` could enclose text with any macros that write to the index in an environment or a `\long` macro defined like:

```
\newcommand{\fix}[1]{\def\textlatin##1{##1}#1}
```

2.5.6 Detecting Punctuation

In Western names, some affixes with full stops could appear at the end of a sentence. Such affixes include “Jr.” (junior), “Sr.” (senior), “d.J.” (*der Jüngere*), and “d.Ä.” (*der Ältere*). Consider this example, where some lines have two full stops and some do not:

Macro Example		Resulting text
<code>\Name[Martin Luther]{King, Jr.}</code>	2 → 1	MARTIN LUTHER KING JR.
<code>\Name[Martin Luther]{King, Jr.}</code>	2 → 1	King.
<code>\Name[Martin Luther]{King, Jr.}</code>	1 → 0	King
<code>\Name*[Martin Luther]{King, Jr.}</code>	2 → 1	Martin Luther King Jr.
<code>\Name*[Martin Luther]{King, Jr.}</code>	1 → 1	Martin Luther King Jr.
<code>{\Name*[Martin Luther]{King, Jr.}}</code>	2 → 2	Martin Luther King Jr.. ⁸

`\Name`, `\FName`, and `\AKA` all check for a trailing full stop in the printed name in the text. If it exists, and if the next token is also a full stop, they gobble the trailing full stop. Grouping tokens, among other items, can frustrate this detection, as shown in the previous example.

⁸Example of how to frustrate the full stop detection mechanism.

2.5.7 Accented Names

For texts that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. Since version 2.1, `nameauth` generally does not require any special treatment for the leading character of the `<SNN>` field. The only exception is when you need to use `\CapThis` under `pdflatex`. That is the case where you substitute `\AccentCapThis` (Section 2.5.3).⁹

Under NFSS and the `utf8` input encoding, the following glyphs generally are available to you:

À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	FIRST USE
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	second use
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß	FIRST USE
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß	second use
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	FIRST USE
à á â ã ä å æ	ç è é ê ë	ì í î ï ð ñ	second use
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	FIRST USE
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	second use
Ǻ ǻ Ǽ Ǿ ǿ ǿ ǿ	Ǿ ǿ ǿ ǿ ǿ ǿ ǿ	Ǿ ǿ ǿ ǿ ǿ ǿ ǿ	FIRST USE
Ǻ ǻ Ǽ Ǿ ǿ ǿ ǿ	Ǿ ǿ ǿ ǿ ǿ ǿ ǿ	Ǿ ǿ ǿ ǿ ǿ ǿ ǿ	second use
IJ iJ L l L l	Ń ń Ņ ņ Œ œ	Ř ř Ť ť	FIRST USE
IJ iJ L l L l	Ń ń Ņ ņ Œ œ	Ř ř Ť ť	second use
Š š Š š Ť ť Ť ť	Ů ů Ů ů	Ž ž Ž ž Ž ž	FIRST USE
Š š Š š Ť ť Ť ť	Ů ů Ů ů	Ž ž Ž ž Ž ž	second use

These glyphs do not cover some of the extended character sets completely. Unicode characters and control sequences are not interchangeable. Instead of getting long, then short references (if they were the same), you only get long ones below:

<code>\Name[Johann]{Andre\"a}</code>	JOHANN ANDREÄ
<code>\Name[Johann]{Andreä}</code>	JOHANN ANDREÄ
<code>\Name{\AE thelred, II}</code>	ÆTHELRED II
<code>\Name{Æthelred, II}</code>	ÆTHELRED II

Additional accents and glyphs can be used with Unicode input, NFSS, `inputenc`, and `fontenc` when using fonts with TS1 glyphs, *e.g.*, `\usepackage{lmodern}` (per the table on pages 455–63 in *The LaTeX Companion*). The example below lets you type, “In Congress, July 4, 1776.”

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

Additionally, `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to generate GHAZĀLI. But be careful! Control sequences like `\=a` fail when using `makeindex`

⁹The root of this problem is shown by this example: `\def\foo#1#2#3\relax{<#1#2><#3>}`. With `\foo abc\relax` you get `<ab><c>`. With `\foo æbc\relax` you get `<æ><bc>`.

Using `xelatex` and `lualatex` always gives you `<ab><c>` and `<æb><c>`, thus avoiding the problem. Using `pdflatex` with `\CapThis` yields `Argument of \UTFviii@two@octets has an extra }`. Many extant word-capitalizing macros and even kernel macros like `\in@... \ifin@` cause errors when used as we do in `nameauth`.

and `gind.ist`, such as with the `ltxdoc` class, because the equal sign is an “actual” character instead of `@`. Using `\index{Gh{\=a}zali}` halts execution. Using `\index{Gh\=azali}` gives an “azali” entry sorted under “Gh” (thanks DAN LUECKING). This issue is not specific to `nameauth` and affects any document where one uses `gind.ist`.

One may use expandable control sequences in names (thanks Robert Schlicht). Also, you can define letters with `\edef` and `\noexpand` to use in names, as some do to “protect” accented letters in names. As of version 2.0 of `nameauth` helpful concerns expressed by PATRICK COUSOT have been addressed.

2.5.8 Index Sorting

The general practice for sorting with `makeindex -s` involves creating your own `.ist` file (pages 659–65 in *The LaTeX Companion*). Otherwise use the following form that works with both `makeindex` and `texindy`:

```
\index{<sortkey>@<actual>}
```

Before version 2.0 of `nameauth`, one had to sort and index a name like JAN ŁUKASIEWICZ by putting it between `\IndexInactive` and `\IndexActive` while creating a manual index entry.

`\PretagName` Fortunately, the current versions of `nameauth` have adopted an easier solution. The syntax of `\PretagName` is like that of `\TagName`:

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

The `\PretagName` macro does not work exactly like the `\TagName` and `\UntagName` macros (see Section 2.8.2 and following). The main differences are:

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

The `\PretagName` macro creates a sort key terminated with the “actual” character, which is `@` by default. Do not include the “actual” character in the pretag. Now, sorting index entries is as simple as:

```
\PretagName[Jan]{Łukasiewicz}{Łukasiewicz, Jan}
\PretagName{Æthelred, II}{Aethelred 2}
```

One need only pretag names once in the preamble. Every time that one refers to Łukasiewicz or Æthelred, the proper index entry will be created. If you create a cross-reference with `\AKA` and you want to pretag it, see Section 2.7.1.

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

You cannot use index tags if the `nameauth` indexing feature is inactive.

This package tries to work with multiple languages and typesetting engines. The following preamble snippet illustrates how that can be done:

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

The following can be used in the text itself to allow for conditional processing that helps one to document work under multiple engines:

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

2.5.9 Custom Formatting

\NamesFormat When formatting is active, **\NamesFormat** is called at the first instance of a name, and at every instance of a name when the **alwaysformat** option is used. Beyond using the package options, one also can redefine **\NamesFormat** to create custom effects. For example, one might change or suppress formatting in all footnotes:

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

This approach will not print the first use of a name in the body text if it already occurred in the footnotes unless one uses **\ForgetName** to force that. This example takes advantage of the deep scoping of **\@makefntext** in order to use a localized **\def** to make a temporary change. The next section shows how one can use a completely independent system of first and subsequent use in the footnotes.

A second example puts the mention of first names in boldface, with additional notations in the margin if possible:

```

\let\oldformat\NamesFormat
\renewcommand*{\NamesFormat}[1]%
  {\textbf{#1}\ifinner\else
   \marginpar{\raggedleft\scriptsize #1}\fi}
\PretagName{Vlad, Tepeş}{Vlad Tepeş}%

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

```

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

The `quote` environment used above permits local changes to `\NamesFormat` so they revert back to the default: VLAD III DRACULA and Vlad III. For references to “Vlad” instead of “Vlad III” one could use `\Name{Vlad, III Dracula}`.¹⁰

2.5.10 Disable Formatting

`\NamesActive` Using the `frontmatter` option deactivates formatting until `\NamesActive` occurs.
`\NamesInactive` Another macro, `\NamesInactive`, will deactivate formatting again. These two macros toggle two independent systems of formatting and first use.

Here we switch to the “front matter” mode with `\NamesInactive`:

```

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

```

Then we switch back to “main matter” mode with `\NamesActive`:

```

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

```

Notice that we have two independent cases of “first use” above. Consider the two “species” of names to be “non-formatted” and “formatted,” intended for front matter and main matter. Yet one could use this, *e.g.*, in footnotes:

```

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

```

¹⁰Do not mix `\Name{Vlad III, Dracula}` with `\Name{Vlad, III Dracula}` or the old syntax, lest errors bite! You would get multiple index entries with `\Name`, unwanted cross-references with `\AKA` and unexpected forms in the text. The simplified interface helps one to avoid this.

2.6 Tweaks: \ForgetName and \SubvertName

Perhaps the easiest way to avoid the “interspecies clashes” above are the two macros presented here. They are meant for tweaking text at or near final draft stage. They affect both front matter and main matter.

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

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

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

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

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

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

2.7 Name Variant Macros

2.7.1 \AKA

\AKA **\AKA** (meaning *also known as*) handles pseudonyms, stage names, *noms de plume*, and so on in order to replace typing manual cross-references in the index:

\Name $\{\text{Jean, sans Peur}\}$ (**\AKA** $\{\text{Jean, sans Peur}\}$ $\{\text{Jean the Fearless}\}$)
was Duke of Burgundy 1404--1419.
JEAN SANS PEUR (Jean the Fearless) was Duke of Burgundy 1404–1419.

Notice that “Jean the Fearless” receives no special formatting. This is intentional, as it reflects the idea of formatting only those names with main index entries. Nevertheless, the reversing and capitalizing mechanisms do work with **\AKA**. The syntax for **\AKA** is:

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

Only the $\langle FNN \rangle$ and $\langle SNN \rangle$ arguments from **\Name** and friends may be cross-referenced. The new syntax allows **\AKA** to cross-reference all name types. Both macros create a cross-reference in the index from the $\langle Alt.\ FNN \rangle$, $\langle Alt.\ SNN \rangle$, and $\langle Alt.\ names \rangle$ fields to a name defined by $\langle FNN \rangle$ and $\langle SNN \rangle$, regardless of whether that name has been used.

Both macros print only the $\langle Alt.\ FNN \rangle$ and $\langle Alt.\ SNN \rangle$ fields in the body text. If the $\langle Alt.\ names \rangle$ field is present, **\AKA** swaps $\langle Alt.\ names \rangle$ with $\langle Alt.\ FNN \rangle$ in the body text. **\AKA*** just prints $\langle Alt.\ names \rangle$ (if present) in the body text. See also Section 2.8.2.

For the following name types, \AKA and \AKA* yield the same results, using BOB HOPE, LOUIS XIV, and LAO-TZU as examples:

\AKA[Bob]{Hope}[Leslie Townes]{Hope}	Leslie Townes Hope
\AKA{Louis, XIV}{Sun King}	Sun King
\AKA{Lao-tzu}{Li, Er}	Li Er

\AKA ignores the $\langle Alt. names \rangle$ field with the names above. The $\langle Alt. names \rangle$ field as part of the cross-reference was envisaged for names like GREGORY I:

\AKA{Gregory, I}{Gregory}[the Great]	Gregory the Great
\AKA*{Gregory, I}{Gregory}[the Great]	the Great

Using \Name*{Gregory, I} ‘‘\AKA*{Gregory, I}{Gregory}[the Great]’’ prints Gregory I “the Great” in the text. The index has a *see* reference from “Gregory the Great” to “Gregory I.”

A few points for \AKA and \AKA* follow:

- With \AKA, [$\langle FNN \rangle$]{ $\langle SNN \rangle$ } is the main name. The cross-reference is [$\langle Alt. FNN \rangle$]{ $\langle Alt. SNN \rangle$ }[$\langle Alt. names \rangle$]. Please do not think that $\langle Alt. FNN \rangle$ belongs to the main name or \AKA will fail.
- \AKA fails with the old syntax: \AKA{Louis}[XIV]{Sun King}. \AKA and \AKA* fail with the old form: \AKA{Gregory}[I]{Gregory}[the Great].
- The $\langle Alt. SNN \rangle$ field uses comma-delimited suffixes.
- The $\langle Alt. names \rangle$ field does not use comma-delimited suffixes.
- Eastern names work: One can refer to LAFCADIO HEARN as KOIZUMI Yakumo: \CapName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}.
- Particles work: Du Cange is the alternate name for CHARLES DU FRESNE, which is capitalized via \CapThis\AKA. See also Section 2.10.2.
- Reversing works, *e.g.* \RevComma: Hope, Leslie Townes.
- The name fields of \PretagName correspond with the [$\langle Alt. FNN \rangle$]{ $\langle Alt. SNN \rangle$ }[$\langle Alt. names \rangle$] fields of \AKA: \AKA{Vlad III, Dracula}{Vlad, Tepeş} corresponds with \PretagName{Vlad, Tepeş}{Vlad Tepes}. It fails with \PretagName{Vlad}[Tepeş]{Vlad Tepes}.

\AKA will not create multiple cross-references. Handle the special case where one moniker applies to multiple people with a manual solution, *e.g.*, “Snellius” for both WILLEBRORD SNEL VAN ROYEN and his son RUDOLPH SNEL VAN ROYEN:

\index{Snellius|see{Snel van Royen, Rudolph; Snel van Royen, Willebrord}}

Cross-references generated by \AKA and \AKA* are meant only to be *see* references, never page entries. See also Section 2.2.4. In certain cases, the alternate name might need to be indexed with page numbers and *see also* references. Do not use \AKA in those cases, rather, consider the following:

- Refer to the person intended, *e.g.*, MAIMONIDES (Moses ben-Maimon):
`\Name{Maimonides} (\AKA{Maimonides}{Moses ben-Maimon})`
- We now have a name and a *see* reference. Now one must refer to the alternate name, *e.g.*, RAMBAM: `\Name{Rambam}`.
- The alternate name must occur before making a cross-reference to the main name, in this case, Maimonides.
- Add `\index{Rambam|seealso{Maimonides}}` at the end of the document to ensure that it is the last entry among the cross-references. Generally, *see also* references follow *see* references in an index entry.¹¹

Even with the new syntax, using `makeindex` may require some manual entries:

```
\index{Doctor Angelicus@\textit{Doctor Angelicus}}%
|see{Thomas Aquinas}}%
\index{Thomas of Aquino|see{Thomas Aquinas}}%
Perhaps the greatest medieval theologian was \Name{Thomas, Aquinas}
(Thomas of Aquino), also known as \textit{Doctor Angelicus}. "Aquinas"
is not a surname.

Perhaps the greatest medieval theologian was THOMAS AQUINAS (Thomas
of Aquino), also known as Doctor Angelicus. "Aquinas" is not a surname.
```

2.7.2 \PName

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

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

Although `\PName` creates an easy shortcut, its drawbacks are many. It only can use the `⟨FNN⟩⟨SNN⟩` form of `\AKA`. It cannot use `\AKA*`. `\PName` really is ill-suited to work with `\CapName`, `\CapThis`, `\RevComma`, `\RevName`, and the related package options. Use it as needed, and *caveat auctor*.

The author determines the name that is indexed (the first name) and the subsequent name that only occurs as a *see* reference. That subsequent name is never shortened in the text. To do that, using the table below, one would type, *e.g.*, `Arouet\IndexName{Voltaire}` or use the Rambam example above. `\PName` can generate the following examples:

<code>\PName[Mark]{Twain}[Samuel L.]{Clemens}</code>	MARK TWAIN (Samuel L. Clemens)
<code>\PName*[Mark]{Twain}[Samuel L.]{Clemens}</code>	Mark Twain (Samuel L. Clemens)
<code>\PName[Mark]{Twain}[Samuel L.]{Clemens}</code>	Twain (Samuel L. Clemens)
<code>\PName{Voltaire}[François-Marie]{Arouet}</code>	VOLTAIRE (François-Marie Arouet)
<code>\PName*{Voltaire}[François-Marie]{Arouet}</code>	Voltaire (François-Marie Arouet)
<code>\PName{Voltaire}[François-Marie]{Arouet}</code>	Voltaire (François-Marie Arouet)

If one used `\PName{William, I}[William]{the Conqueror}` the body text would look right but the index cross-reference would be in error. Medieval and Eastern names are not suited for `\PName`. For them use `\AKA`.

¹¹Different standards exist for punctuating index entries and cross-references. Check with your publisher, style guide, docs for `xindy` and `makeindex`, and <http://tex.stackexchange.com>.

2.8 Indexing Macros

2.8.1 `\IndexName`

`\IndexName` This macro creates an index entry like those created by `\Name` and friends. It prints no text in the body and permits no special formatting. The syntax is:

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

`\IndexName` complies with the new syntax. If $\langle FNN \rangle$ are absent, it indexes $\langle Alternate names \rangle$ as an affix using the old syntax; otherwise it ignores $\langle Alternate names \rangle$. If indexing is switched off (see Section 2.8.5), this macro does nothing. It will not create index entries for names used with `\AKA` as cross-references.

The indexing mechanism in the `nameauth` package follows *Chicago Manual of Style* standards regarding Western names and affixes. Thus the name Chesley B. Sullenberger III becomes “Sullenberger, Chesley B., III” in the index. This formatting only occurs for Western names (where $\langle FNN \rangle$ are present).

2.8.2 `\TagName`

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

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

Tags created by `\TagName` can be helpful in the indexes of history texts. Several features of this package are designed for historical research. Suppose you are working with medieval subject matter. The following macros come in handy:

<code>\TagName{Leo, I}{, pope}</code>	(in the preamble)
<code>\TagName{Gregory, I}{, pope}</code>	
...	
<code>\Name*{Leo, I} \Name*{Gregory, I}</code>	(first references to LEO I and GREGORY I)
...	
<code>\Name*{Leo, I} was known as</code>	Leo I was known as Leo
<code>\AKA{Leo, I}{Leo}[the Great].</code>	the Great.
...	
<code>\Name{Gregory, I} ‘‘\AKA*{Gregory, I}%</code>	Gregory “the Great,” an-
<code>{Gregory}[the Great],’’ another major</code>	other major pope.
<code>pope.</code>	

Here `\TagName` causes the `nameauth` indexing macros to append “`,Lpope`” to the index entries for Gregory I and Leo I.

Tags are literal text that can be daggers, asterisks, and even specials. For example, all fictional names in the index of this manual have an asterisk without any spaces before it. If space is desired between the entry and the tag, one must add the space at the start of the tag. Tagging aids scholarly indexing and can include life/regnal dates and other information.

`\TagName` works with all name types, not just medieval names. Back in Section 2.2 we had the example of Jimmy Carter (cross-reference in the index). `\TagName` adds “`,Lpresident`” to his index entry.

You can use the `{<tag>}` field of `\TagName` to add specials to index entries for names. Every name in this document is tagged with at least `{|hyperpage|}` to allow hyperlinks in the index using the `ltxdoc` class and `hypdoc` package.

2.8.3 \UntagName

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

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

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

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

This refers to JOHN SMITH.                                index: Smith, John
Now another JOHN SMITH.                                    index: Smith, John (second)
Then a third JOHN SMITH.                                    index: Smith, John (third)
Then the first John Smith.                                  index: Smith, John
```

The tweaking macros `\ForgetName` and `\SubvertName` make it seem like you are dealing with three people who have the same name. The index tags will group together those entries with the same tag.¹²

Please remember to note the differences between `\TagName` and `\UntagName` on the one hand and `\PretagName` on the other. See Section 2.5.8.

2.8.4 Global Name Exclusion

`\ExcludeName` This macro globally prevents the indexing of a particular name or cross-reference. If you do not use it at the beginning of the document, you may not exclude any name or cross-reference that has been used already. The syntax is:

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

For example, `\ExcludeName[Kris]{Kringle}` will permit KRIS KRINGLE and Kringle to appear in the body text via `\Name[Kris]{Kringle}`, but no index entry can occur for this name. `\ExcludeName[Santa]{Claus}` will prevent `\AKA[Kris]{Kringle}[Santa]{Claus}` Santa Claus from generating a cross-reference in the index. Instead of the global `\ExcludeName`, it is likelier that you would enclose `\Name`, etc. between `\IndexInactive` and `\IndexActive`.

¹²Since this document, unlike the example above, puts an asterisk by all fictional names in the index, it puts an asterisk at the beginning of the tags above and does not `\UntagName` John Smith, but retags him with an asterisk again.

2.8.5 Indexing Control

`\IndexActive` Using the `noindex` option deactivates the indexing function of this package until `\IndexActive` occurs. Another macro, `\IndexInactive`, will deactivate indexing again. These can be used throughout the document, independently of `\ExcludeName`. They are global in scope, as are the other toggle macros in this package, so one must be explicit in turning indexing on and off.

2.9 Variant Spellings

This section illustrates why this package is called “nameauth.” Here we get to an example where the macros work together to implement a name authority.

Handling variant name spellings can be complicated. For example, let us assume that you are editing a collection of essays. You might settle on the form W.E.B. Du Bois in your name authority. An essay in that collection might use the alternate spelling W.E.B. DuBois. The author or publisher who owns that work might not grant you permission to alter the spelling. In that case, you could add an alternate spelling. Using the simplified interface, it would be:

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

If you wanted to index the alternate spelling with its own entry, the trivial use of `\AltDuBois` allows that easily. All you need do is make cross-references to each variant in the index so that the reader is aware of them.

Nevertheless, Du Bois and DuBois differ only by spaces. For several good reasons, such as fault tolerance in typing, the first/subsequent use mechanism ignores spaces and sees them as *the same name*. Use `\ForgetName[W.E.B.]{Du Bois}` to trigger the first use of `\AltDuBois` in that section.

If you wanted to index the variants under only one name entry, it gets more complicated. You could do the following:

1. Use `\ForgetName[W.E.B.]{Du Bois}` at the start of the section.
2. Wrap `\AltDuBois` between `\IndexInactive` and `\IndexActive`.
3. Call `\IndexName` with the authoritative form right after `\IndexActive`.
4. Create a cross-reference in the index.

This can be automated at the start of the section with something like:

```
\ForgetName[W.E.B.]{DuBois}
\gdef\OtherDuBois{\IndexInactive\AltDuBois\IndexActive%
  \IndexName[W.E.B.]{Du Bois}}
\index{DuBois, W.E.B.|see{Du Bois, W.E.B.}}
```

The alternate section mentions `\OtherDuBois` starting with a first use: W.E.B. DuBOIS. Subsequent uses of `\OtherDuBois` print DuBois. Of course, one could get more complex than the example above. The index will only hold the standard entry for W.E.B. Du Bois: “Du Bois, W.E.B.” and a cross-reference from the variant “DuBois, W.E.B.” to the standard entry.

2.10 Naming Pattern Reference

2.10.1 Basic Naming

Western Names

<i>First reference in the text:</i> JOHN SMITH	<code>\Name*[John]{Smith}</code> <code>\Name[John]{Smith}</code> <code>\FName[John]{Smith}</code>
<i>Subsequent full:</i> John Smith	<code>\Name*[John]{Smith}</code>
<i>Subsequent surname:</i> Smith	<code>\Name[John]{Smith}</code>
<i>Subsequent forename:</i> John	<code>\FName[John]{Smith}</code>
<i>Long first reference:</i> JANE Q. PUBLIC	<code>\Name*[J.Q.]{Public}[Jane Q.]</code> <code>\Name[J.Q.]{Public}[Jane Q.]</code> <code>\FName[J.Q.]{Public}[Jane Q.]</code>
<i>Subsequent full:</i> J.Q. Public	<code>\Name*[J.Q.]{Public}</code>
<i>Alternate:</i> Jane Qetsiyah Public	<code>\Name*[J.Q.]{Public}[Jane Qetsiyah]</code>
<i>Alternate:</i> Janie	<code>\FName[J.Q.]{Public}[Janie]</code>

Western Plus Affixes

Always use a comma to delimit name/affix pairs.

<i>First reference:</i> GEORGE S. PATTON JR.	<code>\Name*[George S.]{Patton, Jr.}</code> <code>\Name[George S.]{Patton, Jr.}</code> <code>\FName[George S.]{Patton, Jr.}</code>
<i>Subsequent:</i> George S. Patton Jr.	<code>\Name*[George S.]{Patton, Jr.}</code>
<i>Subsequent surname:</i> Patton	<code>\Name[George S.]{Patton, Jr.}</code>
<i>Subsequent forename:</i> George	<code>\FName[George S.]{Patton, Jr.}[George]</code>

```
\begin{nameauth}  
  \< Smith & John & Smith & >  
  \< JQP & J.Q. & Public & >  
  \< Patton & George S. & Patton, Jr. & >  
\end{nameauth}
```

```
\Smith, \LSmith, \Smith, and \SSmith:  
  JOHN SMITH, John Smith, Smith, and John  
\JQP[Jane Q.], \LJQP[Jane Q.], and \JQP[Jane Q.]:  
  JANE Q. PUBLIC, Jane Q. Public, and Public  
\LJQP[Jane Qetsiyah]\ and \SJQP[Janie]:  
  Jane Qetsiyah Public and Janie  
\Patton, \LPatton, \Patton, and \SPatton:  
  GEORGE S. PATTON JR., George S. Patton Jr., Patton, and George S.  
\SPatton[George] prints George.
```

New Syntax: Royal, Eastern, and Ancient

Using `\Name{Demetrius, I Soter}` keeps the number with the affix. To keep the number with the name, use `\Name{Demetrius I, Soter}`. See also Section 2.4.1.

<i>First reference:</i> FRANCIS I	<code>\Name*{Francis, I}</code> <code>\Name{Francis, I}</code> <code>\FName{Francis, I}</code>
<i>Subsequent full:</i> Francis I	<code>\Name*{Francis, I}</code>
<i>Subsequent name:</i> Francis	<code>\Name{Francis, I}</code> <code>\FName{Francis, I}</code>
<i>First reference:</i> DEMETRIUS I SOTER	<code>\Name*{Demetrius, I Soter}</code> <code>\Name{Demetrius, I Soter}</code> <code>\FName{Demetrius, I Soter}</code>
<i>Subsequent full:</i> Demetrius I Soter	<code>\Name*{Demetrius, I Soter}</code>
<i>Subsequent name:</i> Demetrius	<code>\Name{Demetrius, I Soter}</code> <code>\FName{Demetrius, I Soter}</code>
<i>First reference:</i> SUN YAT-SEN	<code>\Name*{Sun, Yat-sen}</code> <code>\Name{Sun, Yat-sen}</code> <code>\FName{Sun, Yat-sen}</code>
<i>Subsequent full:</i> Sun Yat-sen	<code>\Name*{Sun, Yat-sen}</code>
<i>Subsequent name:</i> Sun	<code>\Name{Sun, Yat-sen}</code> <code>\FName{Sun, Yat-sen}</code>
<i>First mononym reference:</i> PLATO	<code>\Name*{Plato}</code> <code>\Name{Plato}</code> <code>\FName{Plato}</code>
<i>Subsequent:</i> Plato	<code>\Name*{Plato}</code> <code>\Name{Plato}</code> <code>\FName{Plato}</code>

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

`\Francis`, `\LFrancis`, `\Francis`, and `\SFrancis`:

FRANCIS I, Francis I, Francis, and Francis

`\Dem`, `\LDem`, `\Dem`, and `\SDem`:

DEMETRIUS I SOTER, Demetrius I Soter, Demetrius, and Demetrius

`\Sun`, `\LSun`, `\Sun`, and `\SSun`:

SUN YAT-SEN, Sun Yat-sen, Sun, and Sun

`\Plato`, `\LPlato`, `\Plato`, and `\SPlato`:

PLATO, Plato, Plato, and Plato.

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

Old Syntax: Royal and Eastern

Avoid these forms except with the `comma` option. `\Name{Ptolemy}[I Soter]` keeps the number with the affix. Use `\Name{Ptolemy I}[Soter]` to keep the number with the name. See also Section 2.4.1.

<i>First reference:</i> HENRY VIII	<code>\Name*{Henry}[VIII]</code> <code>\Name{Henry}[VIII]</code> <code>\FName{Henry}[VIII]</code>
<i>Subsequent full:</i> Henry VIII	<code>\Name*{Henry}[VIII]</code>
<i>Subsequent name:</i> Henry	<code>\Name{Henry}[VIII]</code> <code>\FName{Henry}[VIII]</code>
<i>First reference:</i> PTOLEMY I SOTER	<code>\Name*{Ptolemy}[I Soter]</code> <code>\Name{Ptolemy}[I Soter]</code> <code>\FName{Ptolemy}[I Soter]</code>
<i>Subsequent full:</i> Ptolemy I Soter	<code>\Name*{Ptolemy}[I Soter]</code>
<i>Subsequent name:</i> Ptolemy	<code>\Name{Ptolemy}[I Soter]</code> <code>\FName{Ptolemy}[I Soter]</code>
<i>First reference:</i> MAO TSE-TUNG	<code>\Name*{Mao}[Tse-tung]</code> <code>\Name{Mao}[Tse-tung]</code>
<i>Subsequent full:</i> Mao Tse-tung	<code>\Name*{Mao}[Tse-tung]</code>
<i>Subsequent name:</i> Mao	<code>\Name{Mao}[Tse-tung]</code> <code>\FName{Mao}[Tse-tung]</code>

`\begin{nameauth}`

`\< Henry & & Henry & VIII >`
`\< Ptol & & Ptolemy & I Soter >`
`\< Mao & & Mao & Tse-tung >`

`\end{nameauth}`

`\Henry`, `\LHenry`, `\Henry`, and `\SHenry`:

HENRY VIII, Henry VIII, Henry, and Henry

`\Ptol`, `\LPtol`, `\Ptol`, and `\SPtol`:

PTOLEMY I SOTER, Ptolemy I Soter, Ptolemy, and Ptolemy

`\Mao`, `\LMao`, `\Mao`, and `\SMao`:

MAO TSE-TUNG, Mao Tse-tung, Mao, and Mao

Avoid mixing old and new syntax. In the body text, `\Name{Antiochus, IV}` and `\Name{Antiochus, IV}[Epiphanes]` look alike, but their index entries differ.

- Use `\Name{Antiochus, IV Epiphanes}` to get ANTIOCHUS IV EPIPHANES and Antiochus in the text and “Antiochus IV Epiphanes” in the index.
- Use `\Name{Antiochus-IV, Epiphanes}` to get ANTIOCHUS IV EPIPHANES and Antiochus IV in the text and “Antiochus IV Epiphanes” in the index.
- Use `\Name{Antiochus, IV}` to get ANTIOCHUS IV and Antiochus in the text. Use something like `\TagName{Antiochus, IV}{ Epiphanes}` to get “Antiochus IV Epiphanes” in the index and add “Epiphanes” in the text.

2.10.2 Particles

The following illustrate the American style of particulate names.

<i>First:</i> WALTER DE LA MARE	<code>\Name*[Walter]{de la Mare}</code> <code>\Name[Walter]{de la Mare}</code> <code>\FName[Walter]{de la Mare}</code>
<i>Subsequent:</i> de la Mare	<code>\Name[Walter]{de la Mare}</code>
<i>Start of sentence:</i> De la Mare	<code>\CapThis\Name[Walter]{de la Mare}</code>
<i>Forename:</i> Walter	<code>\FName[Walter]{de la Mare}</code>

The Continental style differs slightly. These first three forms below put the particles in the index. Long macros are split for readability.

<i>The (admittedly long) first use:</i> JOHANN WOLFGANG VON GOETHE	<code>\Name*[Johann Wolfgang von]{Goethe}</code> <code>\Name[Johann Wolfgang von]{Goethe}</code> <code>\FName[Johann Wolfgang von]{Goethe}</code>
<i>Subsequent:</i> Goethe	<code>\Name[Johann Wolfgang von]{Goethe}</code>
<i>Forenames:</i> Johann Wolfgang	<code>\FName[Johann Wolfgang von]{Goethe}% [Johann Wolfgang]</code>

These latter examples of the Continental style use the nickname feature to omit the particles from the index.

<i>First:</i> ADOLF VON HARNACK	<code>\Name*[Adolf]{Harnack}[Adolf von]</code> <code>\Name[Adolf]{Harnack}[Adolf von]</code> <code>\FName[Adolf]{Harnack}[Adolf von]</code>
<i>Subsequent full:</i> Adolf von Harnack	<code>\Name*[Adolf]{Harnack}[Adolf von]</code>
<i>Subsequent surname:</i> Harnack	<code>\Name[Adolf]{Harnack}[Adolf von]</code> <code>\Name[Adolf]{Harnack}</code>
<i>Subsequent forename:</i> Adolf	<code>\FName[Adolf]{Harnack}</code>

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

`\DLM\` and `\CapThis\DLM:`

WALTER DE LA MARE and De la Mare.

`\JWG\` and `\JWG:`

JOHANN WOLFGANG VON GOETHE and Goethe.

`\Harnack[Adolf von]\` and `\Harnack:`

ADOLF VON HARNACK and Harnack

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

3 Implementation

3.1 Boolean Values

Affix Commas

The `comma` and `nocomma` options toggle the first value below, while `\ShowComma` toggles the second. Each instance of `\Name` and `\AKA` reset `@nameauth@ShowComma`.

```
1 \newif\if@nameauth@AlwaysComma
2 \newif\if@nameauth@ShowComma
```

Toggle Formatting

This value is toggled with `\NamesActive` and `\NamesInactive` or the `mainmatter` and `frontmatter` options.

```
3 \newif\if@nameauth@DoFormat
```

Indexing

`\IndexActive` and `\IndexInactive`, with the `index` and `noindex` options, toggle the value below.

```
4 \newif\if@nameauth@DoIndex
```

The `pretag` and `nopretag` options toggle the value below.

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

Name Formatting

The next Boolean values govern full name capitalization, name reversing, and name reversing with commas.

```
6 \newif\if@nameauth@AllCaps
7 \newif\if@nameauth@AllThis
8 \newif\if@nameauth@RevAll
9 \newif\if@nameauth@RevThis
10 \newif\if@nameauth@RevAllComma
11 \newif\if@nameauth@RevThisComma
```

`@nameauth@FirstFormat` toggles the formatting of first occurrences of names.

`@nameauth@AlwaysFormat` forces name formatting whenever formatting is active.

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

`@nameauth@FullName` toggles long or short forms in subsequent name uses.

`@nameauth@FirstName` is used when printing only first names. `@nameauth@AltAKA` is toggled by either `\AKA` or `\AKA*` to print a longer or shorter name.

```
14 \newif\if@nameauth@FullName
15 \newif\if@nameauth@FirstName
16 \newif\if@nameauth@AltAKA
```

`\KeepAffix` toggles the value below. Each instance of `\Name` and `\AKA` reset it.

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

This Boolean value is used for detection of affixes and final periods.

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

This Boolean value is triggered by `\CapThis`. Each instance of `\Name` and `\AKA` reset it.

```
19 \newif\if@nameauth@DoCaps
```

This Boolean value is triggered by `\AccentCapThis` to handle special cases of extended Unicode particle caps. Each instance of `\Name` and `\AKA` reset it.

```
20 \newif\if@nameauth@Accent
```


3.2 Package Options

The following package options interact with many of the prior Boolean values. Suppressing and showing commas is set at load time. Most options can be changed with user interface macros. Avoid changing the internal Boolean values directly.

```
21 \def\@nameauth@Actual{@}
22 \DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
23 \DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
24 \DeclareOption{mainmatter}{\@nameauth@DoFormatrue}
25 \DeclareOption{frontmatter}{\@nameauth@DoFormatfalse}
26 \DeclareOption{index}{\@nameauth@DoIndextrue}
27 \DeclareOption{noindex}{\@nameauth@DoIndexfalse}
28 \DeclareOption{pretag}{\@nameauth@Pretagtrue}
29 \DeclareOption{nopretag}{\@nameauth@Pretagfalse}
30 \DeclareOption{allcaps}{\@nameauth@AllCapstrue}
31 \DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
32 \DeclareOption{allreversed}%
33   {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
34 \DeclareOption{allrevcomma}%
35   {\@nameauth@RevAlltrue\@nameauth@RevAllCommatrue}
36 \DeclareOption{notreversed}%
37   {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
38 \DeclareOption{alwaysformat}{\@nameauth@AlwaysFormatrue}
39 \DeclareOption{smallcaps}{\newcommand*{\NamesFormat}{\scshape}}
40 \DeclareOption{italic}{\newcommand*{\NamesFormat}{\itshape}}
41 \DeclareOption{boldface}{\newcommand*{\NamesFormat}{\bfseries}}
42 \DeclareOption{noformat}{\newcommand*{\NamesFormat}{}}
43 \ExecuteOptions%
44   {nocomma,%
45     mainmatter,%
46     index,%
47     pretag,%
48     normalcaps,%
49     notreversed,%
50     smallcaps}
51 \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.

```
52 \RequirePackage{etoolbox}
53 \RequirePackage{ifluatex}
54 \RequirePackage{ifxetex}
55 \RequirePackage{suffix}
56 \RequirePackage{trimspaces}
57 \RequirePackage{xargs}
```

3.3 Internal Macros

<code>\@nameauth@Clean</code>	<p>Thanks to Heiko Oberdiek, this macro produces a “sanitized” string, even using accented characters, based on the parameters of <code>\Name</code> and friends. With this we can construct a control sequence name and test for it to determine the existence of pseudonyms and the first or subsequent occurrences of a name.</p> <pre> 58 \newcommand*{\@nameauth@Clean}[1]% 59 {\expandafter\zap@space\detokenize{#1} \@empty}</pre>
<code>\@nameauth@Root</code>	<p>The following two macros parse $\langle SNN \rangle$ into a radix and a comma-delimited suffix, returning only the radix. They (and their parameters) are expandable in order to facilitate proper indexing functionality. They form the kernel of the suffix removal and comma suppression features.</p> <pre> 60 \newcommand*{\@nameauth@Root}[1]% 61 {\@nameauth@TrimRoot#1,\@empty\relax}</pre>
<code>\@nameauth@TrimRoot</code>	<p>Throw out the comma and suffix, return the radix.</p> <pre> 62 \def\@nameauth@TrimRoot#1,#2\relax{\trim@spaces{#1}}</pre>
<code>\@nameauth@CapRoot</code>	<p>The next two macros implement the particulate name capitalization mechanism by returning a radix where the first letter is capitalized. In <code>xelatex</code> and <code>lualatex</code> this is trivial and causes no problems. In <code>pdflatex</code> we have to account for “double-wide” accented Unicode characters.</p> <pre> 63 \newcommand*{\@nameauth@CapRoot}[1]% 64 {% 65 \ifxetex 66 \@nameauth@CRii#1\relax% 67 \else 68 \ifluatex 69 \@nameauth@CRii#1\relax% 70 \else 71 \if@nameauth@Accent 72 \@nameauth@CRiii#1\relax% 73 \else 74 \@nameauth@CRii#1\relax% 75 \fi 76 \fi 77 \fi 78 }</pre>
<code>\@nameauth@CRii</code>	<p>Grab the first letter as one parameter, and everything before <code>\relax</code> as the second. Capitalize the first and return it with the second.</p> <pre> 79 \def\@nameauth@CRii#1#2\relax{\uppercase{#1}\@nameauth@Root{#2}}</pre>
<code>\@nameauth@CRiii</code>	<p>This is called in <code>pdflatex</code> under <code>inputenc</code> where an accented Unicode character takes the first two parameters. Grab the first “letter” as two parameters and cap it, then everything before <code>\relax</code> as the third. Capitalize the first and return it with the second.</p> <pre> 80 \def\@nameauth@CRiii#1#2#3\relax{\uppercase{#1#2}\@nameauth@Root{#3}}</pre>
<code>\@nameauth@AllCapRoot</code>	<p>This macro returns a fully-capitalized radix. It is used for generating capitalized Eastern family names in the body text.</p> <pre> 81 \newcommand*{\@nameauth@AllCapRoot}[1]% 82 {\uppercase{\@nameauth@Root{#1}}}</pre>

`\@nameauth@Suffix` The following two macros parse $\langle SNN \rangle$ into a radix and a comma-delimited suffix, returning only the suffix. Anything before a comma is stripped off by `\@nameauth@Suffix`, but a comma must be present in the parameter. Leading spaces are removed to allow consistent formatting.

```

83 \newcommand*{\@nameauth@Suffix}[1]%
84   {\@nameauth@TrimSuffix#1\relax}

```

`\@nameauth@TrimSuffix` Throw out the radix, comma, and `\relax`; return the suffix with no leading spaces.

```

85 \def\@nameauth@TrimSuffix#1,#2\relax{\trim@spaces{#2}}

```

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

```

86 \newcommand*{\@nameauth@TestDot}[1]%
87 {%
88   \def\TestDot##1.\TestEnd##2\TestStop{\TestPunct{##2}}%
89   \def\TestPunct##1%
90     {\ifx\TestPunct##1\TestPunct\else\@nameauth@Puncttrue\fi}%
91   \@nameauth@Punctfalse%
92   \TestDot#1\TestEnd.\TestEnd\TestStop%
93 }

```

`\@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.

```

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

```

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

```

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

```

`\@nameauth@FmtName` The following macros format the output of `\Name`, etc. `\@nameauth@FmtName` prints names in the body text, either formatted or not. Notice how `\NamesFormat` (Section 2.5.9) sits between a `\bgroup` and an `\egroup` to localize the font change. `\@nameauth@AlwaysFormat` will force formatting when possible.

```

98 \newcommand*{\@nameauth@FmtName}[1]%
99 {%
100   \if\@nameauth@AlwaysFormat\@nameauth@FirstFormattrue\fi
101   \@nameauth@TestDot{#1}%
102   \if\@nameauth@DoFormat
103     \if\@nameauth@FirstFormat
104       \bgroup\NamesFormat{#1}\egroup%
105     \else
106       #1%
107     \fi
108   \else
109     #1%
110   \fi
111 }

```

```

\@nameauth@Index If the indexing flag is true, create an index entry, otherwise do nothing.
112 \newcommand*{\@nameauth@Index}[2]%
113 {%
114   \def\cseq{#1}%
115   \ifcsname\cseq!TAG\endcsname
116     \ifcsname\cseq!PRE\endcsname
117       \if@nameauth@DoIndex
118         \index{\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
119       \fi
120     \else
121       \if@nameauth@DoIndex\index{#2\csname\cseq!TAG\endcsname}\fi
122     \fi
123   \else
124     \ifcsname\cseq!PRE\endcsname
125       \if@nameauth@DoIndex\index{\csname\cseq!PRE\endcsname#2}\fi
126     \else
127       \if@nameauth@DoIndex\index{#2}\fi
128     \fi
129   \fi
130 }

```

3.4 User Interface Macros

The following macros have been documented previously. Either they call the internal macros or they set Boolean values.

<code>\CapThis</code>	Tells the root capping macro to cap an unaccented character. 131 <code>\newcommand*{\CapThis}{\@nameauth@DoCapstrue}</code>
<code>\AccentCapThis</code>	Tells the root capping macro to cap an accented Unicode character. 132 <code>\newcommand*{\AccentCapThis}{\@nameauth@Accenttrue\@nameauth@DoCapstrue}</code>
<code>\CapName</code>	Capitalize entire name. 133 <code>\newcommand*{\CapName}{\@nameauth@AllThistrue}</code>
<code>\RevName</code>	Reverse name order. 134 <code>\newcommand*{\RevName}{\@nameauth@RevThistrue}</code>
<code>\RevComma</code>	Last name, comma, first name. 135 <code>\newcommand*{\RevComma}%</code> 136 <code>{\@nameauth@RevThistrue\@nameauth@RevThisCommatrue}</code>
<code>\ShowComma</code>	Put comma between name and suffix one time. 137 <code>\newcommand*{\ShowComma}{\@nameauth@ShowCommatrue}</code>
<code>\KeepAffix</code>	Trigger a name-suffix pair to be separated by a non-breaking space. 138 <code>\newcommand*{\KeepAffix}{\@nameauth@NBSPtrue}</code>
<code>\NamesInactive</code>	Switch to the “non-formatted” species of names. 139 <code>\newcommand*{\NamesInactive}{\@nameauth@DoFormatfalse}</code>
<code>\NamesActive</code>	Switch to the “formatted” species of names. 140 <code>\newcommand*{\NamesActive}{\@nameauth@DoFormattrue}</code>
<code>\AllCapsInactive</code>	Turn off global surname capitalization. 141 <code>\newcommand*{\AllCapsInactive}{\@nameauth@AllCapsfalse}</code>
<code>\AllCapsActive</code>	Turn on global surname capitalization. 142 <code>\newcommand*{\AllCapsActive}{\@nameauth@AllCapstrue}</code>
<code>\ReverseInactive</code>	Turn off global name reversing. 143 <code>\newcommand*{\ReverseInactive}{\@nameauth@RevAllfalse}</code>
<code>\ReverseActive</code>	Turn on global name reversing. 144 <code>\newcommand*{\ReverseActive}{\@nameauth@RevAlltrue}</code>
<code>\ReverseCommaInactive</code>	Turn off global “last-name-comma-first.” 145 <code>\newcommand*{\ReverseCommaInactive}%</code> 146 <code>{\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}</code>
<code>\ReverseCommaActive</code>	Turn on global “last-name-comma-first.” 147 <code>\newcommand*{\ReverseCommaActive}%</code> 148 <code>{\@nameauth@RevAlltrue\@nameauth@RevAllCommatrue}</code>
<code>\IndexInactive</code>	turn off global indexing of names. 149 <code>\newcommand*{\IndexInactive}{\@nameauth@DoIndexfalse}</code>

```

\IndexActive turn on global indexing of names.
150 \newcommand*{\IndexActive}{\@nameauth@DoIndextrue}

\IndexActual turn on global indexing of names.
151 \newcommand*{\IndexActual}[1]{\gdef\@nameauth@Actual{#1}}

\Name Here is the heart of the package. Marc van Dongen provided the basic structure. Parsing,
indexing, and formatting are in discrete elements.
152 \newcommandx*\Name[3][1=\@empty, 3=\@empty]%
153 {%
154   \let\ex\expandafter%
155   \leavevmode\hbox{}%

Names occur in horizontal mode; we ensure that. Next we make copies of the arguments
to test them and make parsing decisions
156   \protected@edef\testa{#1}%
157   \protected@edef\arga{\trim@spaces{#1}}%
158   \protected@edef\testb{\trim@spaces{#2}}%
159   \protected@edef\testbr{\@nameauth@Root{#2}}%
160   \protected@edef\testc{#3}%
161   \protected@edef\argc{\trim@spaces{#3}}%
162   \def\csb{\@nameauth@Clean{#2}}%
163   \def\csbc{\@nameauth@Clean{#2#3}}%
164   \def\csab{\@nameauth@Clean{#1!#2}}%

Test for malformed input.
165   \ifx\testb\@empty
166     \PackageError{nameauth}%
167     {macro \Name: Essential name missing}%
168   \else
169     \ifx\csb\@empty
170       \PackageError{nameauth}%
171       {macro \Name: Essential name malformed}%
172     \fi
173   \fi

If global caps. reversing, and commas are true, set the local flags true.
174   \if@nameauth@AllCaps\@nameauth@AllThistrue\fi
175   \if@nameauth@RevAll\@nameauth@RevThistrue\fi
176   \if@nameauth@RevAllComma\@nameauth@RevThisCommatrue\fi

The code below handles non-breaking and regular spaces, as well as commas, in the
text and the index by setting up which kind we want to use. These will be inserted as
appropriate as the output is formatted.
177   \protected@edef\ISpace{\space}%
178   \protected@edef\Space{\space}%
179   \if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
180   \if@nameauth@AlwaysComma
181     \protected@edef\ISpace{,\space}%
182     \protected@edef\Space{,\space}%
183   \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
184   \fi
185   \if@nameauth@ShowComma
186     \protected@edef\ISpace{,\space}%
187     \protected@edef\Space{,\space}%
188   \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
189   \fi

```

The section below parses any “surnames” into name/suffix pairs and figures out how to capitalize and reverse them as needed, storing the results for the main parser.

```

190 \protected@edef\RawShort{\@nameauth@Root{#2}}%
191 \if@nameauth@DoCaps
192 \protected@edef\CapShort{\@nameauth@CapRoot{#2}}%
193 \else
194 \let\CapShort\RawShort%
195 \fi
196 \protected@edef\AllCapShort{\@nameauth@AllCapRoot{#2}}%
197 \let\IndexShort\RawShort%
198 \ifx\testb\testbr
199 \protected@edef\Suff{\@empty}%
200 \let\IndexSNN\RawShort%
201 \let\Reversed\RawShort%
202 \let\SNN\RawShort%
203 \let\PrintShort\RawShort%
204 \if@nameauth@DoCaps
205 \let\Reversed\CapShort%
206 \let\SNN\CapShort%
207 \let\PrintShort\CapShort%
208 \fi
209 \if@nameauth@AllThis
210 \let\Reversed\AllCapShort%
211 \let\SNN\AllCapShort%
212 \let\PrintShort\AllCapShort%
213 \fi
214 \else
215 \protected@edef\Suff{\@nameauth@Suffix{#2}}%
216 \protected@edef\IndexSNN{\RawShort\ISpace\Suff}%
217 \protected@edef\Reversed{\Suff\Space\RawShort}%
218 \protected@edef\SNN{\RawShort\Space\Suff}%
219 \if@nameauth@RevThis
220 \let\PrintShort\Suff%
221 \else
222 \let\PrintShort\RawShort%
223 \fi
224 \if@nameauth@DoCaps
225 \protected@edef\Reversed{\Suff\Space\CapShort}%
226 \protected@edef\SNN{\CapShort\Space\Suff}%
227 \if@nameauth@RevThis
228 \let\PrintShort\Suff%
229 \else
230 \let\PrintShort\CapShort%
231 \fi
232 \fi
233 \if@nameauth@AllThis
234 \protected@edef\Reversed{\Suff\Space\AllCapShort}%
235 \protected@edef\SNN{\AllCapShort\Space\Suff}%
236 \if@nameauth@RevThis
237 \let\PrintShort\Suff%
238 \else
239 \let\PrintShort\AllCapShort%
240 \fi
241 \fi
242 \fi

```

Here we parse names.

```
243 \ifx\testa\@empty
244 \ifx\testc\@empty
```

This is the section for momonyms, royal name/suffix pairs, and native Eastern names where comma-delimited suffixes are used. The first conditional below checks if we are trying to use an alternate name cross-reference as a main name (code !PN for pseudonym). If we are using a legitimate name, we generate an index entry.

```
245 \ifcsname\csb!PN\endcsname
246 \PackageWarning{nameauth}%
247 {macro \Name: Xref: #2 cannot be a page reference}%
248 \else
249 \@nameauth@Index{\csb}{\IndexSNN}%
250 \fi
```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code !MN for main matter name). First we handle subsequent uses. We need `\expandafter` to enable the punctuation detection.

```
251 \if@nameauth@DoFormat
252 \ifcsname\csb!MN\endcsname
253 \if@nameauth@FirstName
254 \@nameauth@FullNamefalse%
255 \@nameauth@FirstNamefalse%
256 \fi
257 \if@nameauth@FullName
258 \@nameauth@FullNamefalse%
259 \if@nameauth@RevThis
260 \ex\@nameauth@FmtName\ex{\Reversed}%
261 \else
262 \ex\@nameauth@FmtName\ex{\SNN}%
263 \fi
264 \else
265 \ex\@nameauth@FmtName\ex{\PrintShort}%
266 \fi
267 \else
```

Handle first uses.

```
268 \@nameauth@FirstFormattrue%
269 \@nameauth@FullNamefalse%
270 \@nameauth@FirstNamefalse%
271 \csgdef{\csb!MN}{}%
272 \if@nameauth@RevThis
273 \ex\@nameauth@FmtName\ex{\Reversed}%
274 \else
275 \ex\@nameauth@FmtName\ex{\SNN}%
276 \fi
277 \fi
278 \else
```


Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```

279      \ifcsname\csb!NF\endcsname
280      \if@nameauth@FirstName
281          \@nameauth@FullNamefalse%
282          \@nameauth@FirstNamefalse%
283      \fi
284      \if@nameauth@FullName
285          \@nameauth@FullNamefalse%
286          \if@nameauth@RevThis
287              \ex\@nameauth@FmtName\ex{\Reversed}%
288          \else
289              \ex\@nameauth@FmtName\ex{\SNN}%
290          \fi
291      \else
292          \ex\@nameauth@FmtName\ex{\PrintShort}%
293      \fi
294  \else

```

Handle first uses.

```

295      \@nameauth@FullNamefalse%
296      \@nameauth@FirstNamefalse%
297      \csgdef{\csb!NF}{}%
298      \if@nameauth@RevThis
299          \ex\@nameauth@FmtName\ex{\Reversed}%
300      \else
301          \ex\@nameauth@FmtName\ex{\SNN}%
302      \fi
303  \fi
304 \fi
305 \else

```

This is the section that handles the old syntax for royal names and native Eastern names. The first conditional below checks if we are trying to use an alternate name cross-reference as a main name (code !PN for pseudonym). If we are using a legitimate name, we generate an index entry.

```

306      \ifcsname\csbc!PN\endcsname
307      \PackageWarning{nameauth}%
308      {macro \Name: Xref: #2 #3 cannot be a page reference}%
309  \else
310      \@nameauth@Index{\csbc}{\IndexSNN\ISpace\argc}%
311  \fi

```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code !MN for main matter name). First we handle subsequent uses.

```

312     \if@nameauth@DoFormat
313         \ifcsname\csbc!MN\endcsname
314             \if@nameauth@FirstName
315                 \@nameauth@FullNamefalse%
316                 \@nameauth@FirstNamefalse%
317             \fi
318             \if@nameauth@FullName
319                 \@nameauth@FullNamefalse%
320                 \if@nameauth@RevThis
321                     \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
322                 \else
323                     \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
324                 \fi
325             \else
326                 \if@nameauth@RevThis
327                     \ex\@nameauth@FmtName\ex{\argc}%
328                 \else
329                     \ex\@nameauth@FmtName\ex{\PrintShort}%
330                 \fi
331             \fi
332         \else

```

Handle first uses.

```

333         \@nameauth@FirstFormattrue%
334         \@nameauth@FullNamefalse%
335         \@nameauth@FirstNamefalse%
336         \csgdef{\csbc!MN}{}%
337         \if@nameauth@RevThis
338             \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
339         \else
340             \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
341         \fi
342     \fi
343 \else

```

Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```

344         \ifcsname\csbc!NF\endcsname
345         \if@nameauth@FirstName
346             \@nameauth@FullNamefalse%
347             \@nameauth@FirstNamefalse%
348         \fi
349         \if@nameauth@FullName
350             \@nameauth@FullNamefalse%
351             \if@nameauth@RevThis
352                 \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
353             \else
354                 \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
355             \fi
356         \else
357             \if@nameauth@RevThis
358                 \ex\@nameauth@FmtName\ex{\argc}%
359             \else
360                 \ex\@nameauth@FmtName\ex{\PrintShort}%
361             \fi
362         \fi
363     \else

```

Handle first uses.

```

364         \@nameauth@FullNamefalse%
365         \@nameauth@FirstNamefalse%
366         \csgdef{\csbc!NF}{}%
367         \if@nameauth@RevThis
368             \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
369         \else
370             \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
371         \fi
372     \fi
373 \fi
374 \fi
375 \else

```

This is the section that handles Western names and non-native Eastern names. The first pair of conditionals handle the `comma` option, `\evThisComma`, and alternate forenames. The next conditional below checks if we are trying to use an alternate name cross-reference as a main name (code `!PN` for pseudonym). If we are using a legitimate name, we generate an index entry.

```

376     \if@nameauth@RevThisComma
377         \protected@edef\ISpace{,\space}%
378         \protected@edef\Space{,\space}%
379         \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
380     \fi
381     \ifx\testc\@empty
382         \let\FNN\arga%
383     \else
384         \let\FNN\argc%
385     \fi
386     \ifcsname\csab!PN\endcsname
387         \PackageWarning{nameauth}%
388         {macro \Name: Xref: #1 #2 cannot be a page reference}%
389     \else
390         \ifx\Suff\@empty
391             \@nameauth@Index{\csab}{\IndexShort,\space\arga}%
392         \else
393             \@nameauth@Index{\csab}{\IndexShort,\space\arga,\space\Suff}%
394         \fi
395     \fi

```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code `!MN` for main matter name). First we handle subsequent uses.

```

396     \if@nameauth@DoFormat
397         \ifcsname\csab!MN\endcsname
398             \if@nameauth@FirstName
399                 \@nameauth@FullNamefalse%
400                 \@nameauth@FirstNamefalse%
401                 \let\PrintShort\FNN%
402             \fi
403             \if@nameauth@FullName
404                 \@nameauth@FullNamefalse%
405                 \if@nameauth@RevThis
406                     \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
407                 \else
408                     \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
409                 \fi
410             \else
411                 \ex\@nameauth@FmtName\ex{\PrintShort}%
412             \fi
413         \else

```

Handle first uses.

```

414      \@nameauth@FirstFormattrue%
415      \@nameauth@FullNamefalse%
416      \@nameauth@FirstNamefalse%
417      \csgdef{\csab!MN}{}%
418      \if@nameauth@RevThis
419        \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
420      \else
421        \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
422      \fi
423    \fi
424  \else

```

Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```

425      \ifcsname\csab!NF\endcsname
426      \if@nameauth@FirstName
427        \@nameauth@FullNamefalse%
428        \@nameauth@FirstNamefalse%
429        \let\PrintShort\FNN%
430      \fi
431      \if@nameauth@FullName
432        \@nameauth@FullNamefalse%
433        \if@nameauth@RevThis
434          \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
435        \else
436          \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
437        \fi
438      \else
439        \ex\@nameauth@FmtName\ex{\PrintShort}%
440      \fi
441    \else

```

Handle first uses.

```

442      \@nameauth@FullNamefalse%
443      \@nameauth@FirstNamefalse%
444      \csgdef{\csab!NF}{}%
445      \if@nameauth@RevThis
446        \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
447      \else
448        \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
449      \fi
450    \fi
451  \fi
452 \fi

```

Reset all the “per name” Boolean values.

```

453 \nameauth@FirstFormatfalse%
454 \nameauth@NBSPfalse%
455 \nameauth@DoCapsfalse%
456 \nameauth@Accentfalse%
457 \nameauth@AllThisfalse%
458 \nameauth@ShowCommafalse%
459 \nameauth@RevThisfalse%
460 \nameauth@RevThisCommafalse%

```

Call the full stop detection.

```

461 \ifnameauth@Punct\expandafter\nameauth@CheckDot\fi
462 }

```

\Name* **\Name*** sets a Boolean value and calls **\Name**.

```

463 \WithSuffix\def\Name*{\nameauth@FullNametrue\Name}

```

\FName **\FName** sets a Boolean value and calls **\Name**.

```

464 \def\FName{\nameauth@FirstNametrue\Name}

```

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

```

465 \WithSuffix\def\FName*{\nameauth@FirstNametrue\Name}

```

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

```

466 \newcommandx*AKA[5][1=@empty, 3=@empty, 5=@empty]%
467 {%
468 \let\ex\expandafter%
469 \leavevmode\hbox{}%

```

Names occur in horizontal mode; we ensure that. Next we make copies of the arguments to test them and make parsing decisions.

```

470 \protected@edef\testa{#1}%
471 \protected@edef\arga{\trim@spaces{#1}}%
472 \protected@edef\testb{\trim@spaces{#2}}%
473 \protected@edef\testbr{\nameauth@Root{#2}}%
474 \protected@edef\testc{#3}%
475 \protected@edef\argc{\trim@spaces{#3}}%
476 \protected@edef\testd{\trim@spaces{#4}}%
477 \protected@edef\testdr{\nameauth@Root{#4}}%
478 \protected@edef\teste{#5}%
479 \protected@edef\arge{\trim@spaces{#5}}%
480 \def\csd{\nameauth@Clean{#4}}%
481 \def\csde{\nameauth@Clean{#4#5}}%
482 \def\csdc{\nameauth@Clean{#3!#4}}%

```

Test for malformed input.

```
483 \ifx\testb\@empty
484   \PackageError{nameauth}%
485     {macro \AKA: Essential name missing}%
486 \else
487   \ifx\csb\@empty
488     \PackageError{nameauth}%
489       {macro \AKA: Essential name malformed}%
490   \fi
491 \fi
492 \ifx\testd\@empty
493   \PackageError{nameauth}%
494     {macro \AKA: Essential name missing}%
495 \else
496   \ifx\csd\@empty
497     \PackageError{nameauth}%
498       {macro \AKA: Essential name malformed}%
499   \fi
500 \fi
```

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

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

The code below handles non-breaking and regular spaces, as well as commas, in the text and the index by setting up which kind we want to use. These will be inserted as appropriate as the output is formatted.

```
504 \protected@edef\ISpace{\space}%
505 \protected@edef\Space{\space}%
506 \if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
507 \if@nameauth@AlwaysComma
508   \protected@edef\ISpace{,\space}%
509   \protected@edef\Space{,\space}%
510   \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
511 \fi
512 \if@nameauth@ShowComma
513   \protected@edef\ISpace{,\space}%
514   \protected@edef\Space{,\space}%
515   \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
516 \fi
```

The section below parses any “surnames” into name/suffix pairs and figures out how to capitalize and reverse them as needed, storing the results for the main parser. We have to handle several more combinations here than with \Name above.

```

517 \protected@edef\Shortb{\@nameauth@Root{#2}}%
518 \protected@edef\Shortd{\@nameauth@Root{#4}}%
519 \if@nameauth@DoCaps
520 \protected@edef\CapShort{\@nameauth@CapRoot{#4}}%
521 \else
522 \let\CapShort\Shortd
523 \fi
524 \protected@edef\AllCapShort{\@nameauth@AllCapRoot{#4}}%
525 \ifx\testb\testbr
526 \let\SNNb\Shortb%
527 \protected@edef\Suffb{\@empty}%
528 \else
529 \protected@edef\Suffb{\@nameauth@Suffix{#2}}%
530 \protected@edef\SNNb{\Shortb\ISpace\Suffb}%
531 \fi
532 \ifx\testd\testdr
533 \protected@edef\Suffd{\@empty}%
534 \let\ISNNd\Shortd%
535 \let\Reversed\Shortd%
536 \let\SNNd\Shortd%
537 \if@nameauth@DoCaps
538 \let\SNNd\CapShort%
539 \let\Reversed\CapShort%
540 \fi
541 \if@nameauth@AllThis
542 \let\SNNd\AllCapShort%
543 \let\Reversed\AllCapShort%
544 \fi
545 \else
546 \protected@edef\Suffd{\@nameauth@Suffix{#4}}%
547 \protected@edef\ISNNd{\Shortd\ISpace\Suffd}%
548 \protected@edef\Reversed{\Suffd\Space\Shortd}%
549 \protected@edef\SNNd{\Shortd\Space\Suffd}%
550 \if@nameauth@DoCaps
551 \protected@edef\Reversed{\Suffd\Space\CapShort}%
552 \protected@edef\SNNd{\CapShort\Space\Suffd}%
553 \fi
554 \if@nameauth@AllThis
555 \protected@edef\Reversed{\Suffd\Space\AllCapShort}%
556 \protected@edef\SNNd{\AllCapShort\Space\Suffd}%
557 \fi
558 \fi

```


Here we parse names.

```
559 \ifx\testc\@empty
560 \ifx\teste\@empty
```

For mononyms and name/suffix pairs: If a pseudonym has not been generated by \AKA or \ExcludeName, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear in the index.

```
561 \ifcsname\csd!PN\endcsname
562 \PackageWarning{nameauth}%
563 {macro \AKA: XRef: #4 exists}%
564 \else
565 \ifcsname\csd!MN\endcsname
566 \PackageWarning{nameauth}%
567 {macro \AKA: Name reference: #4 exists; no xref}%
568 \else
569 \ifcsname\csd!NF\endcsname
570 \PackageWarning{nameauth}%
571 {macro \AKA: Name reference: #4 exists; no xref}%
572 \else
573 \csgdef{\csd!PN}{}%
574 \ifx\testa\@empty
575 \@nameauth@Index{\csd}%
576 {\ISNNd|see{\SNNb}}%
577 \else
578 \ifx\Suffb\@empty
579 \@nameauth@Index{\csd}%
580 {\ISNNd|see{\SNNb,\space\arga}}%
581 \else
582 \@nameauth@Index{\csd}%
583 {\ISNNd|see{\Shortb,\space\arga,\space\Suffb}}%
584 \fi
585 \fi
586 \fi
587 \fi
588 \fi
```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index. Again, \expandafter is used for the punctuation detection.

```
589 \if@nameauth@RevThisComma
590 \protected@edef\ISpace{,\space}%
591 \protected@edef\Space{,\space}%
592 \if@nameauth@NBSP
593 \protected@edef\Space{,\nobreakspace}%
594 \fi
595 \fi
596 \if@nameauth@RevThis
597 \ex\@nameauth@FmtName\ex{\Reversed}%
598 \else
599 \ex\@nameauth@FmtName\ex{\SNNd}%
600 \fi
601 \else
```

For name/affix using the old syntax: If a pseudonym has not been generated by \AKA or \ExcludeName, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear in the index.

```

602     \ifcsname\csde!PN\endcsname
603         \PackageWarning{nameauth}%
604         {macro \AKA: XRef: #4 #5 exists}%
605     \else
606         \ifcsname\csde!MN\endcsname
607             \PackageWarning{nameauth}%
608             {macro \AKA: Name reference: #4 #5 exists; no xref}%
609         \else
610             \ifcsname\csde!NF\endcsname
611                 \PackageWarning{nameauth}%
612                 {macro \AKA: Name reference: #4 #5 exists; no xref}%
613             \else
614                 \csgdef{\csde!PN}{}%
615                 \ifx\testa\@empty
616                     \@nameauth@Index{\csde}%
617                     {\ISNnd\ISpace\arge|see{\SNNb}}%
618                 \else
619                     \ifx\Suffb\@empty
620                         \@nameauth@Index{\csde}%
621                         {\ISNnd\ISpace\arge|see{\SNNb,\space\arga}}%
622                     \else
623                         \@nameauth@Index{\csde}%
624                         {\ISNnd\ISpace\arge|see{\Shortb,\space\arga,\space\Suffb}}%
625                     \fi
626                 \fi
627             \fi
628         \fi
629     \fi

```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index.

```

630     \if@nameauth@RevThisComma
631         \protected@edef\ISpace{\,\space}%
632         \protected@edef\Space{\,\space}%
633     \if@nameauth@NBSP
634         \protected@edef\Space{\,\nobreakspace}%
635     \fi
636 \fi
637 \if@nameauth@AltAKA
638     \ex\@nameauth@FmtName\ex{\arge}%
639 \else
640     \if@nameauth@RevThis
641         \ex\@nameauth@FmtName\ex{\ex\arge\ex\Space\SNNd}%
642     \else
643         \ex\@nameauth@FmtName\ex{\ex\SNNd\ex\space\arge}%
644     \fi
645 \fi
646 \fi
647 \else

```

For Western names and affixes: If a pseudonym has not been generated by \AKA or \ExcludeName, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear in the index.

```

648 \ifcsname\cscd!PN\endcsname
649 \PackageWarning{nameauth}%
650 {macro \AKA: XRef: #3 #4 exists}%
651 \else
652 \ifcsname\cscd!MN\endcsname
653 \PackageWarning{nameauth}%
654 {macro \AKA: Name reference: #3 #4 exists; no xref}%
655 \else
656 \ifcsname\cscd!NF\endcsname
657 \PackageWarning{nameauth}%
658 {macro \AKA: Name reference: #3 #4 exists; no xref}%
659 \else
660 \csgdef{\cscd!PN}{}%
661 \ifx\testa\@empty
662 \ifx\Suffd\@empty
663 \@nameauth@Index{\cscd}%
664 {\ISNND,\space\argc|see{\SNNb}}%
665 \else
666 \@nameauth@Index{\cscd}%
667 {\Shortd,\space\argc,\space\Suffd|see{\SNNb}}%
668 \fi
669 \else
670 \ifx\Suffb\@empty
671 \ifx\Suffd\@empty
672 \@nameauth@Index{\cscd}%
673 {\ISNND,\space\argc|see{\SNNb,\space\arga}}%
674 \else
675 \@nameauth@Index{\cscd}%
676 {\Shortd,\space\argc,\space\Suffd|see{\SNNb,\space\arga}}%
677 \fi
678 \else
679 \ifx\Suffd\@empty
680 \@nameauth@Index{\cscd}%
681 {\ISNND,\space\argc|see{\Shortb,\space\arga,\space\Suffb}}%
682 \else
683 \@nameauth@Index{\cscd}%
684 {\Shortd,\space\argc,\space\Suffd|see{\Shortb,\space\arga,\space\Suffb}}%
685 \fi
686 \fi
687 \fi
688 \fi
689 \fi
690 \fi

```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index.

```

691 \if@nameauth@RevThisComma
692 \protected@edef\ISpace{,\space}%
693 \protected@edef\Space{,\space}%
694 \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
695 \fi
696 \ifx\teste\@empty
697 \let\FNN\argc%
698 \else
699 \let\FNN\arge%
700 \fi
701 \if@nameauth@RevThis
702 \ex\@nameauth@FmtName\ex{\ex\SNNd\ex\Space\FNN}%
703 \else
704 \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNNd}%
705 \fi
706 \fi

```

Reset all the “per name” Boolean values.

```

707 \@nameauth@NBSPfalse%
708 \@nameauth@AltAKAfalse%
709 \@nameauth@DoCapsfalse%
710 \@nameauth@Accentfalse%
711 \@nameauth@AllThisfalse%
712 \@nameauth@ShowCommafalse%
713 \@nameauth@RevThisfalse%
714 \@nameauth@RevThisCommafalse%

```

Call the full stop detection.

```

715 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
716 }

```

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

```

717 \WithSuffix\def\AKA*{\@nameauth@AltAKAtrue\AKA}

```

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

```

718 \newcommand*\PName[5][1=\@empty,3=\@empty,5=\@empty]%
719 {%
720 \Name[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
721 }

```

\PName* This just calls **\Name***, then **\AKA**.

```

722 \WithSuffix\def\PName*{\@nameauth@FullNametrue\PName}

```

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

```

723 \newcommandx*\TagName[4][1=\@empty, 3=\@empty]%
724 {%
725   \protected@edef\testa{#1}%
726   \protected@edef\testb{\trim@spaces{#2}}%
727   \protected@edef\testc{#3}%
728   \def\csb{\@nameauth@Clean{#2}}%
729   \def\csbc{\@nameauth@Clean{#2#3}}%
730   \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them and then we parse the arguments, defining the tag control sequences.

```

731   \ifx\testb\@empty
732     \PackageError{nameauth}%
733     {macro \TagName: Essential name missing}%
734   \else
735     \ifx\csb\@empty
736       \PackageError{nameauth}%
737       {macro \TagName: Essential name malformed}%
738     \fi
739   \fi
740   \ifx\testa\@empty
741     \ifx\testc\@empty
742       \ifcsname\csb!PN\endcsname
743         \PackageWarning{nameauth}%
744         {macro \TagName: not tagging xref: #2}%
745       \else
746         \csgdef{\csb!TAG}{#4}%
747       \fi
748     \else
749       \ifcsname\csbc!PN\endcsname
750         \PackageWarning{nameauth}%
751         {macro \TagName: not tagging xref: #2 #3}%
752       \else
753         \csgdef{\csbc!TAG}{#4}%
754       \fi
755     \fi
756   \else
757     \ifcsname\csab!PN\endcsname
758       \PackageWarning{nameauth}%
759       {macro \TagName: not tagging xref: #1 #2}%
760     \else
761       \csgdef{\csab!TAG}{#4}%
762     \fi
763   \fi
764 }

```

`\UntagName` This deletes an index entry tag.

```
765 \newcommandx*\UntagName[3][1=\@empty, 3=\@empty]%  
766 {%  
767   \protected@edef\testa{#1}%  
768   \protected@edef\testb{\trim@spaces{#2}}%  
769   \protected@edef\testc{#3}%  
770   \def\csb{\@nameauth@Clean{#2}}%  
771   \def\csbc{\@nameauth@Clean{#2#3}}%  
772   \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, undefining the tag control sequences.

```
773   \ifx\testb\@empty  
774     \PackageError{nameauth}%  
775     {macro \UntagName: Essential name missing}%  
776   \else  
777     \ifx\csb\@empty  
778       \PackageError{nameauth}%  
779       {macro \UntagName: Essential name malformed}%  
780     \fi  
781   \fi  
782   \ifx\testa\@empty  
783     \ifx\testc\@empty  
784       \global\csundef{\csb!TAG}%  
785     \else  
786       \global\csundef{\csbc!TAG}%  
787     \fi  
788   \else  
789     \global\csundef{\csab!TAG}%  
790   \fi  
791 }
```

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

```
792 \newcommandx*\PretagName[4][1=\@empty, 3=\@empty]%  
793 {%  
794   \protected@edef\testa{#1}%  
795   \protected@edef\testb{\trim@spaces{#2}}%  
796   \protected@edef\testc{#3}%  
797   \def\csb{\@nameauth@Clean{#2}}%  
798   \def\csbc{\@nameauth@Clean{#2#3}}%  
799   \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, defining the tag control sequences.

```
800   \ifx\testb\@empty  
801     \PackageError{nameauth}%  
802     {macro \TagName: Essential name missing}%  
803   \else  
804     \ifx\csb\@empty  
805       \PackageError{nameauth}%  
806       {macro \TagName: Essential name malformed}%  
807     \fi  
808   \fi  
809   \ifx\testa\@empty  
810     \ifx\testc\@empty  
811       \ifcsname\csb!PN\endcsname  
812         \PackageWarning{nameauth}%  
813         {macro \PretagName: tagging xref: #2}%  
814       \fi  
815       \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi  
816     \else  
817       \ifcsname\csbc!PN\endcsname  
818         \PackageWarning{nameauth}%  
819         {macro \PretagName: tagging xref: #2 #3}%  
820       \fi  
821       \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi  
822     \fi  
823   \else  
824     \ifcsname\csab!PN\endcsname  
825       \PackageWarning{nameauth}%  
826       {macro \PretagName: tagging xref: #1 #2}%  
827     \fi  
828     \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi  
829   \fi  
830 }
```

`\IndexName` This creates an index entry that is not already a pseudonym. It prints nothing. It does ensure consistent formatting.

```

831 \newcommandx*\IndexName[3][1=\@empty, 3=\@empty]%
832 {%
833   \protected@edef\testa{#1}%
834   \protected@edef\arga{\trim@spaces{#1}}%
835   \protected@edef\testb{\trim@spaces{#2}}%
836   \protected@edef\testbr{\@nameauth@Root{#2}}%
837   \protected@edef\testc{#3}%
838   \protected@edef\argc{\trim@spaces{#3}}%
839   \def\csb{\@nameauth@Clean{#2}}%
840   \def\csbc{\@nameauth@Clean{#2#3}}%
841   \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them and make parsing decisions. Below we handle the types of spaces or commas that will be inserted into the index entries.

```

842   \ifx\testb\@empty
843     \PackageError{nameauth}%
844     {macro \IndexName: Essential name missing}%
845   \else
846     \ifx\csb\@empty
847       \PackageError{nameauth}%
848       {macro \IndexName: Essential name malformed}%
849     \fi
850   \fi
851   \protected@edef\Space{\space}%
852   \if@nameauth@AlwaysComma
853     \protected@edef\Space{,\space}%
854   \fi
855   \if@nameauth@ShowComma
856     \protected@edef\Space{,\space}%
857   \fi

```

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

```

858   \let\Short\testbr%
859   \ifx\testb\testbr
860     \let\SNN\Short%
861     \protected@edef\Suff{\@empty}%
862   \else
863     \protected@edef\Suff{\@nameauth@Suffix{#2}}%
864     \protected@edef\SNN{\Short\Space\Suff}%
865   \fi

```


We create the appropriate index entries with tags, letting the internal indexing macro sort that out. We do not create an index entry in the case that a name has been used as a pseudonym by \AKA or \ExcludeName.

```

866 \ifx\testa\@empty
867 \ifx\testc\@empty
868 \ifcsname\csb!PN\endcsname
869 \PackageWarning{nameauth}%
870 {macro \IndexName: XRef: #2 exists}%
871 \else
872 \@nameauth@Index{\csb}{\SNN}%
873 \fi
874 \else
875 \ifcsname\csbc!PN\endcsname
876 \PackageWarning{nameauth}%
877 {macro \IndexName: XRef: #2 #3 exists}%
878 \else
879 \@nameauth@Index{\csbc}{\SNN\Space\argc}%
880 \fi
881 \fi
882 \else
883 \ifcsname\csab!PN\endcsname
884 \PackageWarning{nameauth}%
885 {macro \IndexName: XRef: #1 #2 exists}%
886 \else
887 \ifx\Suff\@empty
888 \@nameauth@Index{\csab}{\Short,\space\arga}%
889 \else
890 \@nameauth@Index{\csab}{\Short,\space\arga,\space\Suff}%
891 \fi
892 \fi
893 \fi
894 \@nameauth@ShowCommfalse%
895 }

```

\ExcludeName This macro prevents a name from being formatted or indexed, making \Name and friends print their arguments, emit a warning, and continue.

```

896 \newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]%
897 {%
898 \protected@edef\testa{#1}%
899 \protected@edef\testb{\trim@spaces{#2}}%
900 \protected@edef\testc{#3}%
901 \def\csb{\@nameauth@Clean{#2}}%
902 \def\csbc{\@nameauth@Clean{#2#3}}%
903 \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them and make parsing decisions. Below we parse the name arguments and create a pseudonym control sequence if it does not exist.

```

904 \ifx\testb\@empty
905 \PackageError{nameauth}%
906 {macro \ExcludeName: Essential name missing}%
907 \else
908 \ifx\csb\@empty
909 \PackageError{nameauth}%
910 {macro \ExcludeName: Essential name malformed}%
911 \fi
912 \fi

```

```

913 \ifx\testa\@empty
914   \ifx\testc\@empty
915     \ifcsname\csb!PN\endcsname
916       \PackageWarning{nameauth}%
917       {macro \ExcludeName: Xref: #2 already exists}%
918     \else
919       \ifcsname\csb!MN\endcsname
920         \PackageWarning{nameauth}%
921         {macro \ExcludeName: Reference: #2 exists; no exclusion}%
922       \else
923         \ifcsname\csb!NF\endcsname
924           \PackageWarning{nameauth}%
925           {macro \ExcludeName: Reference: #2 exists; no exclusion}%
926         \else
927           \csgdef{\csb!PN}{}%
928         \fi
929       \fi
930     \fi
931   \else
932     \ifcsname\csbc!PN\endcsname
933       \PackageWarning{nameauth}%
934       {macro \ExcludeName: Xref: #2 #3 already exists}%
935     \else
936       \ifcsname\csbc!MN\endcsname
937         \PackageWarning{nameauth}%
938         {macro \ExcludeName: Reference: #2 #3 exists; no exclusion}%
939       \else
940         \ifcsname\csbc!NF\endcsname
941           \PackageWarning{nameauth}%
942           {macro \ExcludeName: Reference: #2 #3 exists; no exclusion}%
943         \else
944           \csgdef{\csbc!PN}{}%
945         \fi
946       \fi
947     \fi
948   \fi
949 \else
950   \ifcsname\csab!PN\endcsname
951     \PackageWarning{nameauth}%
952     {macro \ExcludeName: XRef: #1 #2 already exists}%
953   \else
954     \ifcsname\csab!MN\endcsname
955       \PackageWarning{nameauth}%
956       {macro \ExcludeName: Reference: #1 #2 exists; no exclusion}%
957     \else
958       \ifcsname\csab!NF\endcsname
959         \PackageWarning{nameauth}%
960         {macro \ExcludeName: Reference: #1 #2 exists; no exclusion}%
961       \else
962         \csgdef{\csab!PN}{}%
963       \fi
964     \fi
965   \fi
966 \fi
967 }

```

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

```
968 \newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]%
969 {%
970   \protected@edef\testa{#1}%
971   \protected@edef\testb{\trim@spaces{#2}}%
972   \protected@edef\testc{#3}%
973   \def\csb{\@nameauth@Clean{#2}}%
974   \def\csbc{\@nameauth@Clean{#2#3}}%
975   \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, undefining the control sequences.

```
976   \ifx\testb\@empty
977     \PackageError{nameauth}%
978     {macro \ForgetName: Essential name missing}%
979   \else
980     \ifx\csb\@empty
981       \PackageError{nameauth}%
982       {macro \ForgetName: Essential name malformed}%
983     \fi
984   \fi
985   \ifx\testa\@empty
986     \ifx\testc\@empty
987       \global\csundef{\csb!MN}%
988       \global\csundef{\csb!NF}%
989     \else
990       \global\csundef{\csbc!MN}%
991       \global\csundef{\csbc!NF}%
992     \fi
993   \else
994     \global\csundef{\csab!MN}%
995     \global\csundef{\csab!NF}%
996   \fi
997 }
```

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

```
998 \newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]%  
999 {%  
1000 \protected@edef\testa{#1}%  
1001 \protected@edef\testb{\trim@spaces{#2}}%  
1002 \protected@edef\testc{#3}%  
1003 \def\csb{\@nameauth@Clean{#2}}%  
1004 \def\csbc{\@nameauth@Clean{#2#3}}%  
1005 \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, defining the control sequences.

```
1006 \ifx\testb\@empty  
1007 \PackageError{nameauth}%  
1008 {macro \SubvertName: Essential name missing}%  
1009 \else  
1010 \ifx\csb\@empty  
1011 \PackageError{nameauth}%  
1012 {macro \SubvertName: Essential name malformed}%  
1013 \fi  
1014 \fi  
1015 \ifx\testa\@empty  
1016 \ifx\testc\@empty  
1017 \csgdef{\csb!MN}{}%  
1018 \csgdef{\csb!NF}{}%  
1019 \else  
1020 \csgdef{\csbc!MN}{}%  
1021 \csgdef{\csbc!NF}{}%  
1022 \fi  
1023 \else  
1024 \csgdef{\csab!MN}{}%  
1025 \csgdef{\csab!NF}{}%  
1026 \fi  
1027 }
```

nameauth The **nameauth** environment provides a means to implement shorthand references to names in a document.

```

1028 \newenvironment{nameauth}{%
1029   \begingroup%
1030   \let\ex\expandafter%
1031   \csdef{<}&##1&##2&##3&##4>{%
1032     \protected@edef\arga{\trim@spaces{##1}}%
1033     \protected@edef\testb{\trim@spaces{##2}}%
1034     \protected@edef\testc{\trim@spaces{##3}}%
1035     \protected@edef\testd{\trim@spaces{##4}}%
1036     \newtoks\tokb%
1037     \newtoks\tokc%
1038     \newtoks\tokd%
1039     \tokb\expandafter{##2}%
1040     \tokc\expandafter{##3}%
1041     \tokd\expandafter{##4}%
1042     \ifx\arga\@empty
1043       \PackageError{nameauth}%
1044       {environment nameauth: Control sequence missing}%
1045     \else
1046       \ifx\testc\@empty
1047         \PackageError{nameauth}%
1048         {environment nameauth: Essential name missing}%
1049       \else
1050         \ifcsname\arga\endcsname
1051           \PackageWarning{nameauth}%
1052           {environment nameauth: Redefinition of shorthands}%
1053         \fi
1054         \ifx\testd\@empty
1055           \ifx\testb\@empty
1056             \ex\csgdef\ex{\ex\arga\ex}\ex{\ex\Name\ex{\the\tokc}}%
1057             \ex\csgdef\ex{\ex L\ex\arga\ex}\ex{\ex\Name\ex*\ex{\the\tokc}}%
1058             \ex\csgdef\ex{\ex S\ex\arga\ex}\ex{\ex\FName\ex{\the\tokc}}%
1059           \else
1060             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\arga\ex\ex\ex}%
1061             \ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex[\ex\the\ex\tokb\ex]}%
1062             \ex{\the\tokc}}%
1063             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\arga%
1064             \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex*%
1065             \ex\ex\ex[\ex\the\ex\tokb\ex]\ex{\the\tokc}}%
1066             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\arga%
1067             \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\FName\ex\ex\ex[%
1068             \ex\the\ex\tokb\ex]\ex{\the\tokc}}%
1069           \fi
1070         \else
1071           \ifx\testb\@empty
1072             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\arga\ex\ex\ex}%
1073             \ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex{\ex\the\ex\tokc\ex}%
1074             \ex[\the\tokd]}%
1075             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\arga%
1076             \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\Name%
1077             \ex\ex\ex*\ex\ex\ex{\ex\the\ex\tokc\ex}\ex[\the\tokd]}%
1078             \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\arga%
1079             \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\FName\ex\ex\ex{%
1080             \ex\the\ex\tokc\ex}\ex[\the\tokd]}%
1081           \else

```


4 Change History

v0.7		v1.0	
General: Initial release	1	General: Works fully with microtype and memoir	1
v0.75		v1.1	
\ForgetName: New parameter added	58	General: Bugfixes	1
\IndexName: Use current parameters	56	v1.2	
v0.8		\TagName: Added	53
General: Add features, bugfixes	1	\UntagName: Added	54
v0.85		v1.26	
\@nameauth@FmtName: Add comma suppression	35	\@nameauth@CRii: Fixed	34
General: Add package options	1	\AKA: Fix sorting of name suffixes	46
\AKA: Add comma suppression	46	\IndexName: Fix name suffix sorting	56
\IndexName: Add comma suppression	56	v1.4	
\Name: Add comma suppression	38	\@nameauth@Root: Made more robust	34
\PName: Add comma suppression	52	General: Add features, bugfixes	1
\PName*: Add comma suppression	52	\FName: Refactored	46
v0.86		\FName*: Refactored	46
General: Fix regressions	1	\Name*: Refactored	46
v0.9		\ShowComma: Added	37
\@nameauth@Suffix: Added	35	v1.5	
\@nameauth@TrimRoot: Suffix handling expandable	34	\@nameauth@AllCapRoot: Added	34
\@nameauth@TrimSuffix: Added	35	\@nameauth@TrimSuffix: Trim spaces	35
General: Add first name formatting; affix handling expandable	1	General: Add features, bugfixes, options	1
\AKA: Add starred mode; redesigned	46	\AKA: Add reversing and caps	46
\AKA*: Added	52	\AllCapsActive: Added	37
\FName: Added	46	\AllCapsInactive: Added	37
\SubvertName: Added	60	\CapName: Added	37
v0.94		\Name: Add reversing and caps	38
\@nameauth@FmtName: Add particle caps	35	\RevComma: Added	37
\@nameauth@Index: Added	36	\ReverseActive: Added	37
General: Add index suppression, error checking, name particle caps	1	\ReverseCommaActive: Added	37
\CapThis: Added	37	\ReverseCommaInactive: Added	37
\ExcludeName: Added	57	\ReverseInactive: Added	37
\IndexActive: Added	38	\RevName: Added	37
\IndexInactive: Added	37	v1.6	
v0.95		nameauth: Added	61
\@nameauth@CRii: Added	34	v1.7	
\@nameauth@CapRoot: Added	34	General: Fix options processing	1
\@nameauth@FmtName: Works with microtype	35	v1.8	
General: Bugfixes	1	General: Update docs	1
v0.96		v1.9	
General: Bugfixes	1	\ForgetName: Ensure global undef	58
\Name: Works w/ microtype, memoir	38	\KeepAffix: Added	37
		\TagName: Fix cs collisions	53
		\UntagName: Ensure global undef, fix cs collisions	54
		v2.0	
		\@nameauth@FmtName: One macro instead of two	35

\@nameauth@Index: Redesigned	trim spaces; redesign tagging .	38
tagging	\PretagName: Added	55
\@nameauth@TrimRoot: trim spaces	\SubvertName: Isolate malformed	
General: Use dtxgen template in-	input	60
stead of dtxut; update docs . .	\TagName: Isolate malformed input;	
\AKA: Isolate malformed input; trim	redesign tagging	53
spaces; redesign tagging	\UntagName: Isolate malformed in-	
nameauth: Redesigned argument	put; redesign tagging	54
handling	v2.1	
\ExcludeName: Isolate malformed	\@nameauth@CRiii: added	34
input	\@nameauth@CapRoot: Handle Uni-	
\ForgetName: Isolate malformed in-	code better	34
put	General: Isolate Unicode issues . . .	1
\IndexActual: Added	\AccentCapThis: Added	37
\IndexName: Isolate malformed in-	\AKA: Isolate Unicode issues	46
put; trim spaces; redesign tag-	\Name: Isolate Unicode issues . . .	38
ging	v2.11	
\Name: Isolate malformed input;	nameauth: Bugfix	61

5 Index

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

Symbols		
<code>\@nameauth@AllCapRoot</code>	Chiang Kai-shek, presi- dent	Hope, Bob 6, 23
. 81	Cicero, M.T. 12, 13	Hope, Leslie Townes <i>see</i> Hope, Bob
<code>\@nameauth@CRii</code> 79	Clemens, Samuel L. <i>see</i> Twain, Mark	I
<code>\@nameauth@CRiii</code> 80	Confucius 12, 13	<code>\IndexActive</code> 27, 150
<code>\@nameauth@CapRoot</code> 63	Cousot, Patrick 19	<code>\IndexActual</code> 19, 151
<code>\@nameauth@CheckDot</code> 94	D	<code>\IndexInactive</code> 27, 149
<code>\@nameauth@Clean</code> 58	Dagobert I, king 9	<code>\IndexName</code> 25, 831
<code>\@nameauth@EvalDot</code> 96	de la Mare, Walter 16, 31	Iron Mike <i>see</i> Tyson, Mike
<code>\@nameauth@FmtName</code> 98	de Soto, Hernando 8	Ishida Yoko 15
<code>\@nameauth@Index</code> 112	Demetrius I Soter, king 29	J
<code>\@nameauth@Root</code> 60	<i>Doctor Angelicus</i> <i>see</i> Thomas Aquinas	Jean sans Peur, duke 22
<code>\@nameauth@Suffix</code> 83	Dongen, Marc van 2, 38	Jean the Fearless <i>see</i> Jean sans Peur
<code>\@nameauth@TestDot</code> 86	Du Bois, W.E.B. 27	John Eriugena 16
<code>\@nameauth@TrimRoot</code> 62	du Cange <i>see</i>	Just Boris <i>see</i>
<code>\@nameauth@TrimSuffix</code> 85	du Fresne, Charles	Boris the Animal
A	du Fresne, Charles 23	K
<code>\AccentCapThis</code> 16, 132	DuBois, W.E.B. <i>see</i> Du Bois, W.E.B.	Kanno, Yoko† 15
<code>\Ethelred II, king</code> 18, 19	E	<code>\KeepAffix</code> 14, 138
<code>\AKA</code> 22, 466	Einstein, Albert 12, 13	King, Martin Luther, Jr. 16, 17
<code>\AKA*</code> 22, 717	Elizabeth I, queen 7, 8	Koizumi Yakumo <i>see</i> Hearn, Lafcadio
<code>\AllCapsActive</code> 15, 142	environments:	Konoe, Fumimaro, PM† 8
<code>\AllCapsInactive</code> 15, 141	<code>nameauth</code> 7, 1028	Kresge, Joseph <i>see</i> Kreskin, J.
Anaximander 16	<code>\ExcludeName</code> 26, 896	Kreskin, J. 10
Andreaä, Johann 18	F	L
Antiochus IV	<code>\FName</code> 13, 464	Lao-tzu 23
Epiphanes, king 30	<code>\FName*</code> 13, 465	Leo I, pope 25
Arai Akino 15	<code>\ForgetName</code> 22, 968	Leo the Great <i>see</i> Leo I
Aristotle 7, 8	Francis I, king 29	Li Er <i>see</i> Lao-tzu
Arouet, François-Marie <i>see</i> Voltaire	G	Louis XIV, king 14, 23
Atila the Hun 8	Goethe, Johann Wolf- gang von 31	Lueck, Uwe 2, 35
B	Gossett, Louis, Jr. 14	Luecking, Dan 19
Biermann, Johann* 17	Gregorio, Enrico 2	Łukasiewicz, Jan 19
Boris the Animal 9	Gregory I, pope 23, 25	M
C	Gregory the Great <i>see</i> Gregory I	Maimonides 24
<code>\CapName</code> 15, 133	H	Malebranche, Nicolas 21
<code>\CapThis</code> 16, 131	Hammerstein, Oskar, II 14, 16	Mao Tse-tung, chair- man 16, 30
Carnap, Rudolph 21	Harnack, Adolf 31	Mill, J.S. 16
Carter, James Earl, Jr., president 4, 25	Hearn, Lafcadio 23	Moses ben-Maimon <i>see</i> Maimonides
Carter, Jimmy <i>see</i> Carter,	Henry VIII, king 9, 14, 30	
James Earl, Jr.		
Charles the Bald, em- peror 12, 13		

N		\ReverseInactive 15 , 143		Sun Yat-sen, president
Nakano, Aiko†	15	\RevName	15 , 134	14 , 29
\Name	12 , 152	Rockefeller, Jay		
\Name*	12 , 463	. <i>see</i> Rockefeller,		
nameauth (environ-		John David, IV		
ment)	7 , 1028	Rockefeller, John David,		
\NamesActive	21 , 140	II	6 , 8	
\NamesFormat	20	Rockefeller, John David,		
\NamesInactive	21 , 139	IV	6 , 8 , 13	
Nogawa Sakura	15	S		
O		Schlicht, Robert	2 , 19	
Oberdiek, Heiko	2 , 34	Shikata Akiko	15	
P		\ShowComma	14 , 137	
Patton, George S., Jr.	28	Smith, John*	26 , 28	
Plato	29	Smith, John* (other)	26	
\PName	24 , 718	Smith, John* (third)	26	
\PName*	722	Snel van Royen,		
\PretagName	19 , 792	Rudolph	23	
Ptolemy I Soter, king	30	Snel van Royen, Wille-		
Public, J.Q.*	28	brord	23	
R		Snellius . <i>see</i> Snel van		
Rambam	24 ,	Royen, Rudolph;		
<i>see also</i> Maimonides		Snel van		
\RevComma	16 , 135	Royen, Willebrord		
\ReverseActive	15 , 144	Stephani, Philipp	2	
\ReverseCommaActive	16 , 147	Strietelmeier, John	17	
\ReverseCommaInactive	16 , 145	\SubvertName	22 , 998	
		Sullenberger, Chesley		
		B., III	13 , 25	
		Sun King . <i>see</i> Louis XIV		
T		U		
\TagName	25 , 723	\UntagName	26 , 765	
Thomas Aquinas	24	V		
Thomas of Aquino	<i>see</i> Thomas Aquinas	Vlad II Dracul	21	
Twain, Mark	24	Vlad III Dracula	21	
Tyson, Mike	10	Vlad Țepeș	<i>see</i>	
U		Vlad III Dracula		
		Vlad the Impaler	<i>see</i>	
		Vlad III Dracula		
		Voltaire	24	
W		Y		
Washington, George,	6 , 8	Yamamoto Isoroku	7 , 8	
president		Yohko	15	
White, E.B.	10	Yoshida Shigeru, PM	9	