glossaries-extra.sty v2.1: an extension to the glossaries package

Nicola L.C. Talbot

Dickimaw Books
dickimaw-books.com

2026-04-21

This document is also available as HTML (glossaries-extra-manual.html).

Abstract

The glossaries−extra package is an extension to the glossaries package, providing additional features. In particular, it provides a completely different abbreviation mechanism. You will need at least glossaries version 4.19, but it is best to update both packages at the same time, if new releases are available for both of them.

The glossaries−extra package uses a different set of defaults to the base glossaries package. This means that if you simply replace glossaries with glossaries−extra in an existing document, there may be some differences in the PDF, and you may encounter errors. See §1.1 for more details.

This document assumes some familiarity with the glossaries package. If you are new to glossaries, you may prefer to start with the following:

- The glossaries package: a guide for beginners

```
texdoc glossariesbegin
```

- glossaries-extra and bib2gls: an introductory guide

```
texdoc bib2gls-begin
```

Contents

List of Examples

If an example shows the icon then the source code is embedded in the PDF as an attachment. If your PDF viewer supports attachments, you can extract the self-contained example file to try it out for yourself. Alternatively, you can click on the download icon which will try downloading the example source code from your closest CTAN mirror, but make sure that this user manual matches the version on CTAN first. You can also try using:

```
texdoc -l glossaries-extra-manual-example⟨nnn⟩
```

where ⟨*nnn*⟩ is the example number zero-padded to three digits to find out if the example files are installed on your device.

Part I.

User Guide

1. Introduction

The glossaries package is a flexible package, but it's also a heavy-weight package that uses a lot of resources. As package developer, I'm caught between those users who complain about the drawbacks of a heavy-weight package with a large user manual and those users who want more features (which necessarily adds to the package weight and manual size).

The glossaries−extra package is an attempt to provide a compromise for this conflict. Version 4.22 of the glossaries package is the last version to incorporate any major new features. Future versions of glossaries will mostly just be bug fixes. New features will instead be added to glossaries-extra. This means that the base glossaries package won't increase in terms of package loading time and allocation of resources, but those users who do want extra features available will have more of a chance of getting their feature requests accepted.

Some of the commands provided by the base glossaries package are incompatible with glossaries−extra. These are marked with ⊘ in this document.

The glossaries−extra package internally loads the glossaries package. As a general rule, it's better to defer loading the base glossaries package to glossaries−extra rather than loading the two packages separately.

1.1. Package Defaults

I'm not happy with some of the default settings assumed by the glossaries package, and, judging from code I've seen, other users also seem unhappy with them, as certain package options are often used in questions posted on various sites. I can't change the default behaviour of glossaries as it would break backward compatibility, but since glossaries−extra is a separate package, I have decided to implement some of these commonly-used options by default. You can switch them back if they're not appropriate.

The new defaults are:

- `toc=true` (add the glossaries to the table of contents). Use `toc=false` to switch this back off.
- `nopostdot=true` (suppress the terminating full stop after the description in the glossary). Use `nopostdot=false` or just `postdot` to restore the terminating full stop. Alternatively, if you are interested in switching to `bib2gls`, you can instruct `bib2gls` to insert it with the `post-description-dot` option.

- noredefwarn (suppress the warnings that occur when the theglossary environment and \printglossary are redefined while glossaries is loading). Note that this won't have any effect if the glossaries package has already been loaded before you load the glossaries−extra package.
- If babel has been loaded, the translate=babel option is switched on. To revert to using the translator interface, use translate=true. There is no change to the default if babel hasn't been loaded.
- The default style used by \newacronym is short−nolong. (That is, the long form is not shown on first use.) To revert back to "⟨*long*⟩ (⟨*short*⟩)" on first use do:

```
\setabbreviationstyle[acronym]{long-short}
```

In the above example, long−short refers to the glossaries−extra abbreviation style not the glossaries acronym style of the same name. See §4 for further details.

1.2. Example Differences Between glossaries and glossaries−extra

The examples below illustrate the difference in explicit package options between glossaries and glossaries−extra. There may be other differences resulting from modifications to commands provided by glossaries.

1.2.1. Basic defaults

```
\documentclass{article}
\usepackage{glossaries-extra}
```

This is essentially equivalent to:

```
\documentclass{article}
\usepackage[toc,nopostdot]{glossaries}
\usepackage{glossaries-extra}
```

The defaults for glossaries are toc=false and nopostdot=false but for glossaries−extra the defaults are toc=true and nopostdot=true. However, glossaries−extra won't change the settings if glossaries has already been loaded. For example:

```
\documentclass{article}
\usepackage{glossaries}
\usepackage{glossaries-extra}
```

This is now like:

```
\documentclass{article}
\usepackage[toc=false,nopostdot=false]
 {glossaries-extra}
```

In general, it's best to only load glossaries−extra and allow it to implicitly load glossaries. This ensures better integration.

1.2.2. Language defaults

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage{glossaries-extra}
```

This is like:

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage[toc,nopostdot,translate=babel]
{glossaries}
\usepackage{glossaries-extra}
```

By default, the base glossaries package will load translator if babel is detected, but the glossaries −extra won't. Remember that if glossaries has already been loaded before glossaries−extra then it's too late to specify the `translate` option with glossaries−extra as the localisation support will have already been established when glossaries was loaded.

1.2.3. Combined with memoir

```
\documentclass{memoir}
\usepackage{glossaries-extra}
```

This is like:

```
\documentclass{memoir}
\usepackage[toc,nopostdot,noredefwarn]{glossaries}
\usepackage{glossaries-extra}
```

However

```
\documentclass{memoir}
\usepackage{glossaries}
\usepackage{glossaries-extra}
```

This is like:

```
\documentclass{memoir}
\usepackage[toc=false,nopostdot=false]{glossaries}
\usepackage{glossaries-extra}
```

Since by the time glossaries-extra has been loaded, the base glossaries package has already redefined memoir's glossary-related commands.

1.2.4. Abbreviations

Abbreviations are defined with \newabbreviation:

```
\usepackage{glossaries-extra}
\newabbreviation{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form:
\glsxtrfull{svm}.
\end{document}
```

This is the closest match to:

```
\usepackage{glossaries}
\newacronym{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form:
\acrfull{svm}.
\end{document}
```

If you want to continue using `\newacronym` then you will need to change the style for the `acronym` category:

```
\usepackage{glossaries-extra}
\setabbreviationstyle[acronym]{long-short}
\newacronym{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form:
\glsxtrfull{svm}.
\end{document}
```

Don't use commands like `\glsfirst` or `\glstext` with abbreviations. See §4 for further details.

1.2.5. Glossary Mid-Build Placeholder (`\printglossary`)

Another noticeable change with glossaries-extra is that by default `\printglossary` will now display information text in the document if the external glossary file doesn't exist. This is explanatory text to help new users who can't work out what to do next to complete the document build. Once the document is set up correctly and the external files have been generated, this text will disappear.

This change is mostly likely to be noticed by users with one or more redundant empty glossaries who ignore transcript messages, explicitly use `makeindex`/`xindy` on just the non-empty glossary (or glossaries) and use the iterative `\printglossaries` command instead of `\printglossary`. For example, consider the following:

```
\documentclass{article}
\usepackage[acronym]{glossaries}
\makeglossaries
\newacronym{laser}{laser}
```

```
{light amplification by stimulated
emission of radiation}
\begin{document}
\gls{laser}
\printglossaries
\end{document}
```

The above document will only display the list of acronyms at the place where `\printglossaries` occurs. However it will also attempt to input the `gls` file associated with the `main` glossary.

If you use `makeglossaries`, you'll get the warning message:

```
Warning: File 'test.glo' is empty.
Have you used any entries defined in glossary
'main'?
Remember to use package option 'nomain' if you
don't want to use the main glossary.
```

(where the original file is called `test.tex`) but if you simply call `makeindex` directly to generate the `acr` file (without attempting to create the `gls` file) then the transcript file will always contain the message:

```
No file test.gls.
```

This doesn't occur with `makeglossaries` as it will create the `gls` file containing the single command `\null`.

If you simply change from glossaries to glossaries−extra in this document, you'll find a change in the resulting PDF if you don't use `makeglossaries` and you only generate the `acr` file with `makeindex`.

The transcript file will still contain the message about the missing `gls`, but now you'll also see information in the actual PDF document. The simplest remedy is to follow the advice inserted into the document at that point, which is to add the `nomain` package option:

```
\documentclass{article}
\usepackage[nomain,acronym,postdot]
 {glossaries-extra}
\makeglossaries
\setabbreviationstyle[acronym]{long-short}
\newacronym{laser}{laser}
{light amplification by stimulated
```

```
emission of radiation}
\begin{document}
\gls{laser}
\printglossaries
\end{document}
```

Note the need to set the acronym style using `\setabbreviationstyle` before `\newacronym`. See §4 for further details.

1.3. Further Reading

The following documents and web pages are also available:

- The glossaries-extra documented code

```
texdoc glossaries-extra-code
```

- Gallery: glossaries, glossaries-extra and `bib2gls`.[1]
- FAQs: glossaries, glossaries-extra and `bib2gls`.[2]
- Incorporating `makeglossaries` or `makeglossaries-lite` or `bib2gls` into the document build.[3]
- The `bib2gls` application.[4]
- The glossaries package.[5]

[1] dickimaw-books.com/gallery
[2] dickimaw-books.com/faq.php
[3] dickimaw-books.com/latex/buildglossaries/
[4] ctan.org/pkg/bib2gls
[5] ctan.org/pkg/glossaries

2. Package Options

```
\usepackage[⟨options⟩]{glossaries-extra}
```

This chapter describes the package options provided by glossaries−extra that are either not defined by the base glossaries package or are modified by glossaries−extra. You can additionally pass the base package options to glossaries−extra. For example, instead of:

```
\usepackage[nonumberlist]{glossaries}
\usepackage[abbreviations]{glossaries-extra}
```

you can simply do:

```
\usepackage[abbreviations,nonumberlist]
 {glossaries-extra}
```

It's better not to load the glossaries package first. Leave glossaries−extra to load it, where possible, to allow for a smoother integration between the two packages.

After glossaries−extra has been loaded, some of the glossaries−extra package options may be changed with:

```
\glossariesextrasetup{⟨options⟩}
```

where ⟨*options*⟩ are the same as the relevant package option.

Certain options can only be supplied as package options since the settings need to be known while glossaries−extra is loading.

To change the base glossaries package's options (that may be changed after the package has loaded), continue to use:

\setupglossaries{⟨options⟩}

but don't use any of the options listed here in that command.

2.1. Glossary Lists

nomissingglstext=⟨boolean⟩ *default:* **true**; *initial:* **false**

If true, this will suppress the warning written to the transcript and the warning text that will appear in the document if the external glossary files haven't been generated due to an incomplete document build. However, it's probably simpler just to fix whatever has caused the failure to build the external file or files.

abbreviations

This option has no value and can't be cancelled. If used, it will automatically create a new glossary with the label `abbreviations` and redefines `\glsxtrabbrvtype` to this label. (The file extensions are `glg-abr`, `gls-abr` and `glo-abr`.) In addition, this option defines a shortcut command:

\printabbreviations[⟨options⟩]

which is equivalent to:

```
\printglossary[type=\glsxtrabbrvtype,⟨options⟩]
```

If glossaries-extra-bib2gls is also loaded then this option will additionally provide `\printunsrtabbreviations` which uses `\printunsrtglossary` instead.

The title of the new glossary is given by

\abbreviationsname *initial:* Abbreviations

If this command is already defined, it's left unchanged. Otherwise it's defined to "Abbreviations" if babel hasn't been loaded or `\acronymname` if babel has been loaded. However, if you're using babel it's likely you will need to change this. (See §15 for further details.)

If you don't use the abbreviations package option, the \abbreviationsname command won't be defined (unless it's defined by an included language file).

If the abbreviations option is used and the acronym option provided by the glossaries package hasn't been used, then \acronymtype will be set to \glsxtrabbrvtype so that acronyms defined with \newacronym can be added to the list of abbreviations. If you want acronyms in the main glossary and other abbreviations in the abbreviations glossary then you will need to redefine \acronymtype to main:

```
\renewcommand*{\acronymtype}{main}
```

Note that there are no analogous options to the glossaries package's acronymlists option (or associated commands) as the abbreviation mechanism is handled differently with glossaries-extra.

symbols

This is passed to the base glossaries package, but glossaries-extra will additionally define:

```
\glsxtrnewsymbol[⟨key=value list⟩]{⟨entry-label⟩}{⟨sym⟩}
```

which is equivalent to:

```
\newglossaryentry{⟨entry-label⟩}{name={⟨symbol⟩},
 sort={⟨entry-label⟩},type={symbols},category={symbol},
⟨options⟩}
```

Note that the sort key is set to the ⟨*entry-label*⟩ not the ⟨*symbol*⟩ as the symbol will likely contain commands. If this isn't appropriate, you can override it by using the sort key in the optional argument.

This option also sets the regular attribute to true for the symbol category and provides the category post-description hook:

```
\glsxtrpostdescsymbol                                   initial: empty
```

If glossaries-extra-bib2gls is also loaded then this option will additionally provide \printunsrtsymbols which uses \printunsrtglossary.

numbers

This is passed to the base glossaries package but glossaries-extra will additionally define:

\glsxtrnewnumber[⟨key=value list⟩]{⟨entry-label⟩}{⟨num⟩}

which is equivalent to:

```
\newglossaryentry{⟨entry-label⟩}{name={⟨number⟩},
 sort={⟨entry-label⟩},type={numbers},category={number},
⟨options⟩}
```

Note that the sort key is set to the ⟨entry-label⟩. If this isn't appropriate, you can override it by using the sort key in the optional argument.

This option also sets the regular attribute to true for the number category and provides the category post-description hook:

\glsxtrpostdescnumber *initial: empty*

If glossaries-extra-bib2gls is also loaded then this option will additionally provide \printunsrtnumbers which uses \printunsrtglossary.

acronyms

This is passed to the base glossaries package (which defines \printacronyms and creates a new glossary with the label acronym) but if glossaries-extra-bib2gls is loaded then this option will additionally provide \printunsrtacronyms which uses \printunsrtglossary.

As with the base glossaries package, this option redefines \acronymtype to acronym. Note that this option doesn't change \glsxtrabbrvtype.

acronym=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If acronym=true, this behaves like acronyms. Note that acronym=false won't work if the base glossaries package was loaded before glossaries-extra.

index

This is passed to the base glossaries package but if glossaries-extra-bib2gls is loaded then this option will additionally provide \printunsrtindex which uses \printunsrtglossary.

The base package index option also defines:

\newterm[⟨key=value list⟩]{⟨entry-label⟩}

This definition is modified by glossaries-extra to additionally set the category to index and sets the description to discard the post-description hook (\nopostdesc) but retain \glsxtrpostdescription so that the category post-description hook can still be applied.

This option also sets the regular attribute to true for the index category and defines an associated category post-description hook:

\glsxtrpostdescindex *initial: empty*

2.2. Glossary Style Options

nopostdot=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

This option is provided by glossaries where it simply alters a corresponding conditional that's used inside \glspostdescription to determine whether or not to insert a full stop. The nopostdot option is modified by glossaries-extra to reduce interference from the postpunc option (described below).

The nopostdot option will have no effect if the glossary style doesn't include \glspostdescription. (Use stylemods to ensure that all the predefined styles that show the description have this hook added.)

If you are using bib2gls, you may prefer to use the post-description-dot resource option. If you do use that option, make sure that you have nopostdot=true (or postpunc=none) to prevent a double-dot.

The post-description-dot resource option appends a dot to the end of the description value (if set). This means that the post-description hook will be placed after the dot. This is different from using nopostdot=false (or postdot or postpunc) which will append the punctuation mark at the end of the post-description hook.

postdot

This is a shortcut for nopostdot=false.

postpunc=⟨*value*⟩

This option redefines \glspostdescription to display the required punctuation. Note that this means the hook will no longer check for the nopostdot conditional.

This option will have no effect if the glossary style doesn't include \glspostdescription. (Use stylemods to ensure that all the predefined styles that show the description have this hook added.)

The postpunc value may either be the required punctuation or one of the following keywords:

postpunc=dot

This redefines \glspostdescription to use a full stop but also adjusts the space factor. This isn't exactly the same as nopostdot=false since it removes the conditional from \glspostdescription. If you are using bib2gls, you may prefer to use the post-description-dot resource option.

postpunc=comma

This redefines \glspostdescription to a comma.

postpunc=none

This redefines \glspostdescription to do nothing. This isn't exactly the same as nopostdot=true since it removes the conditional from \glspostdescription.

stylemods=⟨*value*⟩ *default:* **default**

Loads the glossaries-extra-stylemods package (see §8.6.5), which patches the styles provided with the base glossaries package so that they all use \glspostdescription. Extra hooks are also provided to make them easier to customize. The value may be one of the following:

stylemods=all

Loads all styles that are provided by both glossaries and glossaries-extra.

stylemods=default

Patches all the predefined styles that have been loaded, without loading any extra styles. This will typically be all the styles that are usually loaded by glossaries (for example, list and long). Package options such as `nolist` will alter which styles are loaded. In the case of `nostyles`, no styles will be loaded, so none of them will be patched.

It's pointless using both `stylemods=default` and `nostyles`. Any glossary style packages that are subsequently loaded won't be patched.

stylemods=⟨*list*⟩

For each element ⟨*tag*⟩ in ⟨*list*⟩, the corresponding package glossary-⟨*tag*⟩ will be loaded. You can use this in combination with `nostyles` to only load the particular style package or packages that you require (without loading the full set of defaults). For example,

```
\usepackage[nostyles,
 stylemods={bookindex,longextra},
 style=bookindex]{glossaries-extra}
```

This prevents the base glossaries package from loading the default set of styles, but loads glossaries-extra-stylemods, glossary-bookindex and glossary-longextra, and then sets the default style to bookindex.

2.3. Loading Other Packages

Some options listed in other sections, such as the `stylemods` and `record` options, also load supplementary packages.

prefix

Loads the glossaries-prefix package (if not already loaded).

accsupp

Loads the glossaries-accsupp package (if not already loaded). This option can only be used as a package option (not in `\glossariesextrasetup`) as glossaries-extra needs to know whether or not to provide accessibility support while it's loading.

The glossaries-accsupp package is still experimental and so accessibility features are liable to change.

If you want to define styles that can interface with the accessibility support provided by glossaries-accsupp use the `\glsaccess`⟨*xxx*⟩ type of commands instead of `\glsentry`⟨*xxx*⟩ (for example, `\glsaccesstext` instead of `\glsentrytext`). If glossaries-accsupp hasn't been loaded those commands are equivalent (for example, `\glsaccesstext` just does `\glsentrytext`) but if it has been loaded, then the `\glsaccess`⟨*xxx*⟩ commands will add the accessibility information. See §9 for further details.

2.4. Entry Definitions, References and Indexing

undefaction=⟨*value*⟩ *initial:* **error**

This indicates what to do if an undefined glossary entry is referenced.

Undefined entries can't be picked up by any commands that iterate over a glossary list. This includes `\forglsentries` and `\glsaddall`.

undefaction=error

Produces an error message for undefined glossary entries.

undefaction=warn

Only produces a warning message for undefined glossary entries. The place where the entry has been referenced will be marked with `??` (as with undefined labels or citations). The unknown marker is produced with:

`\glsxtrundeftag` *initial:* `??`

This defaults to two question marks.

Note that \ifglsused will only display ?? in the document text with undefaction=warn if the entry hasn't been defined, as the underlying boolean variable doesn't exist and so is neither true nor false. (There will also be a warning in the transcript.) You may prefer to use \GlsXtrIfUnusedOrUndefined instead. See §5.10 for further details.

If you want to write a custom command that needs to generate a warning or error for an undefined reference, you can use:

\glsxtrundefaction{⟨*message*⟩}{⟨*additional help*⟩}

This will produce the unknown marker if used within the document environment. Depending on the undefaction, \glsxtrundefaction will either create an error with the given ⟨*message*⟩ and ⟨*additional help*⟩ or will create a warning with the given ⟨*message*⟩.

docdef=⟨*value*⟩ *default:* **true**; *initial:* **false**

This setting governs where \newglossaryentry can be used (preamble-only or anywhere before the first glossary or anywhere within the document).

Commands like \newabbreviation and \glsxtrnewsymbol that internally use \newglossaryentry are also governed by this option. Other commands, such as \longnewglossaryentry are always preamble-only.

With just the base glossaries package, \newglossaryentry is allowed in the document environment as long as you haven't used \makenoidxglossaries. There are, however, problems that can occur when entries are defined within the document environment (see the glossaries documentation for further details). To encourage preamble-only use, the glossaries-extra package prohibits the use of \newglossaryentry within the document environment by default, but if you really want this you can use this package option to allow it.

Note that in the case of bib2gls, all entry data is originally defined in bib files. The entry definitions (using commands like \longnewglossaryentry and \newabbreviation) are written to the glstex files that are input in the preamble.

docdef=false

Prohibits the use of \newglossaryentry within the document environment. All entries must be defined in the preamble.

docdef=true

Permits the use of \newglossaryentry in the document environment provided \makenoidxglossaries hasn't been used (as per the base glossaries package). This will create

a temporary glsdefs file that contains the entry definitions so that they can be available on the next LaTeX run at the beginning of the document to allow any glossaries in the front matter to display correctly.

If all your glossaries occur at the end of the document, consider using docdef=restricted instead.

docdef=restricted

Permits the use of \newglossaryentry in the document environment provided the entry definitions all occur before the first glossary is displayed.

This avoids the need for the glsdefs file. You will still need to take care about any changes made to the category code of characters that are required by the ⟨*key*⟩=⟨*value*⟩ mechanism (that is, the comma and equal sign) and any makeindex or xindy special character that occurs in the sort key or label. If any of those characters are made active in the document (for example, through babel shortcuts), then it can cause problems with the entry definition.

This option will allow \newglossaryentry to be used in the document with \makenoidxglossaries, but note that \longnewglossaryentry remains a preamble-only command.

With this option, if an entry appears in the glossary before it has been defined, an error will occur (or a warning if the undefaction=warn option is used). If you edit your document and either remove an entry or change its label, you may need to delete the document's temporary files (such as the aux and gls files).

docdef=atom

This option behaves like docdef=restricted but creates the glsdefs file for atom's autocomplete support. This file isn't input by glossaries-extra and so associated problems with the use of this file are avoided, but it allows the autocomplete support to find the labels in the file.

A bug fix in glossaries v4.47 has changed the format of the glsdefs file slightly.

As with docdef=restricted, entries may be defined in the preamble or anywhere in the document, but they may only be referenced after they have been defined. Entries must be defined before the associated glossary is displayed.

If you need a list of all entry labels for the use of an editor or helper script you may also want to consider the package options writeglslabels and writeglslabelnames provided by the base glossaries package. Note that with these options and with docdef=atom, only the entry labels that are visible to LaTeX can be saved. So if you are using bib2gls you will only get the labels of the entries that have already been selected by bib2gls. The bib files can be found by parsing the aux file for \glsxtr@resource (listed in the src option or \jobname.bib if src is missing).

shortcuts={⟨*value*⟩} *initial:* **none**

Unlike the base glossaries package option of the same name, this option isn't boolean but has multiple values.

Multiple invocations of the `shortcuts` option *within the same option list* will override each other. Since these options define commands, the action can't be undone with a later `\glossariesextrasetup`.

shortcuts=ac

Set the shortcut commands provided by the base glossaries package for acronyms (such as `\ac`) but use the glossaries-extra abbreviation commands, such as `\glsxtrshort` and `\glsxtrlong`, instead of the analogous base commands, such as `\acrshort` and `\acrlong`. See §4.3.2 for further details.

shortcuts=abbreviations

Sets the abbreviation shortcuts (see §4.3.2). This setting doesn't switch on the acronym shortcuts provided by the base glossaries package.

shortcuts=abbr *alias:* **abbreviations**

Synonym for `shortcuts=abbreviations`.

shortcuts=other

Implements the other (non-abbreviation) shortcut commands:

`\newentry{`⟨*entry-label*⟩`}{`⟨*options*⟩`}`

A synonym for `\newglossaryentry`.

`\newsym[`⟨*key=value list*⟩`]{`⟨*entry-label*⟩`}{`⟨*sym*⟩`}`

A synonym for `\glsxtrnewsymbol` (provided that the `symbols` package option is also used).

\newnum[⟨key=value list⟩]{⟨entry-label⟩}{⟨num⟩}

A synonym for \glsxtrnewnumber (provided that the numbers package option is also used).

shortcuts=acother

Implements shortcuts=ac and shortcuts=other.

shortcuts=abother

Implements shortcuts=abbreviations and shortcuts=other.

shortcuts=all

Implements shortcuts=ac, shortcuts=abbreviations and shortcuts=other.

shortcuts=acronyms

Sets the shortcuts provided by the base glossaries package for acronyms (such as \ac). See the glossaries package documentation for further details.

Note that the short and long forms (\acs and \acl) don't use \glsxtrshort and \glsxtrlong but use the original \acrshort and \acrlong, which aren't compatible with the glossaries–extra abbreviation mechanism. The better option is to use shortcuts=ac.

Don't use shortcuts=acronyms unless you have reverted \newacronym back to the base glossaries package's acronym mechanism. See §4.6 for further details.

shortcuts=acro *alias:* **acronyms**

Synonym for shortcuts=acronyms.

shortcuts=true *alias:* **all**

This setting is provided by the base glossaries package. With glossaries–extra it's equivalent to shortcuts=all.

shortcuts=false *alias:* **all**

This setting is provided by the base glossaries package. With glossaries-extra it's equivalent to shortcuts=none.

indexcrossrefs=⟨*boolean*⟩ *default:* **true**; *initial:* ***varies***

This is a boolean option that governs whether or not to automatically index any cross-referenced entries that haven't been marked as used at the end of the document. These are entries that are identified in one of the cross-referencing fields (see and seealso) of another used entry as opposed to entries that have the cross-referencing fields set.

Since entries with the alias key are intended as synonyms for another term, the target is expected to be indexed so entries with the alias key set aren't affected by this option.

For example:

```
\newglossaryentry{courgette}{name={courgette},
 description={small vegetable marrow}}
\newglossaryentry{marrow}{name={marrow},
 description={long gourd with green skin},
 seealso={courgette}}
```

Suppose that "marrow" is indexed (so that it appears in the glossary with the cross-reference to "courgette") but if courgette isn't indexed anywhere in the document (using commands like \gls or \glsadd) then there will be a broken cross-reference in the marrow location list pointing to courgette, which doesn't appear in the glossary. With indexcrossrefs=true, the courgette entry will be indexed at the end of the document using \glsadd with format=glsxtrunusedformat, which corresponds to the command \glsxtrunusedformat.

Note that this special format \glsxtrunusedformat simply does \unskip and ignores its argument, which creates a blank location. If any of the cross-referenced entries have been indexed but haven't been marked as used (for example, with \glsadd) then this will cause a spurious comma in the location list. This is a limitation of the way that makeindex and xindy work as they are general purpose indexing applications which require locations. If you have entries with cross-references, you may want to consider switching to bib2gls instead.

Note that bib2gls can automatically find dependent entries when it parses the bib source file, so the record option automatically implements indexcrossrefs=false.

This function is implemented by code added to the end document hook that determines whether

or not to use the command \glsxtraddallcrossrefs. This command iterates over all entries in all glossaries, which adds to the overall document build time, especially if you have defined a large number of entries, so this defaults to indexcrossrefs=false, but it will be automatically switched on if you use the see or seealso keys in any entries. See also §5.9.

indexcrossrefs=true

Enables this setting.

indexcrossrefs=false

Disables this setting even if the see or seealso key is present in any entries.

autoseeindex=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

This is a boolean option that governs whether or not the see and seealso keys should automatically index the cross-reference when an entry is defined (see §5.9).

With the base glossaries package, the see key was provided as a shortcut for \glssee. For example:

```
\newglossaryentry{courgette}{name={courgette},
 description={small vegetable marrow}}
\newglossaryentry{zucchini}{name={zucchini},
 description={},
 see={courgette}}
```

is equivalent to:

```
\newglossaryentry{courgette}{name={courgette},
 description={small vegetable marrow}}
\newglossaryentry{zucchini}{name={zucchini},
 description={}}
\glssee{zucchini}{courgette}
```

This was designed for documents where only entries that are actually used in the document are defined and ensures that the cross-reference is included in the glossary, even though it may not be referenced anywhere in the document. However, it becomes problematic if neither entry is required in the document.

The glossaries-extra package modifies the action of the see key so that it also saves the value and will only perform the automated \glssee if autoseeindex=true. Similarly for

the seealso key.

Note that the record option automatically implements autoseeindex=false as the corresponding action can be implemented with bib2gls's selection option.

autoseeindex=true

Enables automatic indexing using \glssee for the see key (as per the base glossaries package) and \glsxtrindexseealso for the seealso key.

For example, if an entry is defined as

```
\newglossaryentry{foo}{name={foo},
 description={},see={bar,baz}}
```

then, with autoseeindex=true and the default indexcrossrefs setting, this is equivalent to

```
\newglossaryentry{foo}{name={foo},description={}}
\glssee{foo}{bar,baz}
\glossariesextrasetup{indexcrossrefs=true}
\GlsXtrSetField{foo}{see}{bar,baz}
```

autoseeindex=false

The value of the see and seealso keys will be stored in their corresponding fields (and can be accessed using commands like \glsxtrusesee and \glsxtruseseealso) but the cross-reference won't be automatically indexed.

Note that indexcrossrefs isn't automatically implemented by the presence of the see key when autoseeindex is false.

For example, if an entry is defined as

```
\newglossaryentry{foo}{name={foo},
 description={},see={bar,baz}}
```

then, with autoseeindex=false and the default indexcrossrefs setting, this is equivalent to

```
\newglossaryentry{foo}{name={foo},
 description={}}
\GlsXtrSetField{foo}{see}{bar,baz}
```

It's therefore possible with this option to remove the cross-references from the location lists and set their position within the glossary style.

Another method of preventing the automatic indexing is to define the entries before the external indexing files have been opened with \makeglossaries. Since the appropriate file isn't open, the information can't be written to it. This will need the package option seenoindex=ignore to prevent an error occurring.

record=⟨*value*⟩ *default:* **only**; *initial:* **off**

This setting indicates whether or not bib2gls is required.

This option can only be set in the preamble and can't be used after \GlsXtrLoadResources or \glsbibdata.

With the record setting on (that is, any value other than off), then any of the commands that would typically index the entry (such as \gls, \glstext or \glsadd) will add a record to the aux file. bib2gls can then read this information to find out which entries have been used. (Remember that commands like \glsentryname don't index, so any use of these commands won't add a corresponding record.) See §11 for further details.

The hybrid method additionally performs the standard indexing action that's required for makeindex or xindy to work, but this can't be done until bib2gls has created the glstex files that provide the entry definitions. In general, it's best to avoid the hybrid method.

record=off

Indexing is performed as per the base glossaries package using either \makeglossaries or \makenoidxglossaries. This setting implements undefaction=error.

record=only

Indexing (or recording) is performed by adding bib2gls records in the aux file. Neither \makeglossaries nor \makenoidxglossaries is permitted. Use \GlsXtrLoadResources (or \glsbibdata) to set up bib2gls resource options. Glossaries should be displayed with the "unsrt" family of commands, such as \printunsrtglos-

sary.

This setting implements undefaction=warn, autoseeindex=false, indexcrossrefs=false sort=none, and automatically loads the supplementary glossaries-extra-bib2gls package. (There should be no need to explicitly load glossaries-extra-bib2gls.)

This option also defines the location and group keys that are set by bib2gls to provide the location list and group information required by the "unsrt" family of commands.

The document build process is (assuming the file is called myDoc.tex):

```
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
```

If you want letter groups you will need the --group or -g switch when invoking bib2gls:

```
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
```

Note that this setting will prevent the see key from automatically implementing \glssee. (bib2gls deals with the see field.) You may explicitly use \glssee in the document, but bib2gls will ignore the cross-reference if the see field was already set for that entry.

record=nameref

As record=only but uses nameref records, which include the current label information given by \@currentlabel and \@currentHref. This means that the title can be included in the entry locations, if available. This setting also supports location hypertargets that don't follow a simple ⟨*h-prefix*⟩⟨*the-counter*⟩ format, which can't be used with other indexing options.

See §11.6.6 for further details of this option.

This option requires hyperref, otherwise it will fall back on the usual location records.

Note that \@currentHref is always globally updated whenever \refstepcounter is used, but \@currentlabel isn't. This can cause some undesired side-effects with some settings. Remember also that the indexcounter option increments the associated counter every time an entry is indexed, which affects this option. If the location counter is the default

page, only the location number is shown.

record=alsoindex *alias:* **hybrid**

Deprecated synonym of record=hybrid.

record=hybrid

This is a hybrid setting that uses bib2gls to fetch entry information from bib files, but uses makeindex or xindy to create the glossary files (which are input with \printglossary). Note that this requires a slower and more complicated build process (see below).

This hybrid approach is provided for the rare instances where an existing xindy rule or module is too complicated to convert to a bib2gls rule but the entries need to be fetched from a bib file. There's no benefit in using this option with makeindex.

This setting does not load glossaries-extra-bib2gls, as bib2gls is only being used to fetch the entry definitions.

Since it's redundant to make bib2gls also sort and collate locations (in addition to xindy performing these tasks), use the resource options sort=none and save-locations=false for a faster build. Many of the other resource options are likely to be irrelevant.

This setting must be used with \makeglossaries but not with its optional argument. Each glossary should be displayed using \printglossary (or \printglossaries for all of them).

This setting should not be used with \makenoidxglossaries.

You may need to change the transcript file used by bib2gls to avoid a clash with xindy's transcript file. This can be done with bib2gls's --log-file or -t option.

The document build process is (assuming the file is called myDoc.tex):

```
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
```

Note that, in this case, it's redundant to call bib2gls with the --group or -g switch as xindy will insert the group heading information into the corresponding glossary file.

If you want bib2gls to form the letter groups then this hybrid method is inappropriate.

bibglsaux=⟨*basename*⟩ *initial:* ***empty***
requires bib2gls v3.0+

Alternatively, this setting can be implemented with:

```
\glsxtrsetbibglsaux{⟨basename⟩}
```

This option should only be used once. If used again no new file will be created. If the ⟨*basename*⟩ value is empty, records will be written to the normal aux file.

A document containing many records can result in a large aux file with information that's only relevant to bib2gls. This option will create a new file called ⟨*basename*⟩.aux that will be used to store the records. The file will be skipped by LaTeX but will be picked up by bib2gls v3.0+ when it inputs the main aux file. Note that this creates an extra write register.

You should still supply the main aux file when you run bib2gls as ⟨*basename*⟩.aux will only contain the records and not the other information that bib2gls requires (such as the resource options).

equations=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

This setting will cause the default location counter to automatically switch to equation when inside a numbered equation environment, such as equation or align. The counter can be explicitly overridden with the counter \glslink option.

floats=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

This setting will cause the default location counter to automatically switch to the corresponding counter when inside a floating environment, such as figure or table. The counter can be explicitly overridden with the counter \glslink option.

Remember that within floats it's the \caption command that actually uses \refstepcounter, so indexing before the caption will result in the wrong reference. The commands for use in captions and sections, such as \glsfmttext and \glsfmtshort, don't index. (See §5.3). You may want to consider using \glsadd after the caption (not before). For example:

```
\begin{figure}[htbp]
  \centering
  \includegraphics{example-image}
  \caption{Sample \glsfmttext{foobar} figure}
  \glsadd{foobar}
\end{figure}
```

indexcounter

This option defines the indexing counter:

wrglossary

which is incremented every time an entry is indexed. This option automatically implements counter=wrglossary. This means that each location will link to the relevant part of the page where the indexing occurred (instead of to the top of the page). See the bib2gls documentation for the save-index-counter resource option for more details.

This option is primarily intended for use with bib2gls (v1.4+) and hyperref. It can be used with makeindex or xindy, but it will interfere with the location list collation, so you won't have ranges and you'll have duplicate page numbers present.

This option works by incrementing wrglossary with \refstepcounter and adding \label. This can cause a problem if the indexing occurs in an equation environment as amsmath forbids multiple occurrences of \label (resulting in the "Multiple \label's" error). It's best to change the counter to page or equation when in maths mode with this option. For example:

```
\renewcommand{\glslinkpresetkeys}{%
 \ifmmode \setupglslink{counter=page}\fi}
\renewcommand{\glsaddpresetkeys}{%
 \ifmmode \setupglsadd{counter=page}\fi}
```

2.5. Debugging

debug=⟨*value*⟩ *default:* **true**; *initial:* **false**

Enables debugging information for draft documents. This option is defined by the base glossaries package, but is extended by glossaries-extra to provide additional settings. If no value is provided, `true` is assumed. The following values are available:

debug=false

This setting is provided by the glossaries package and is the default. This switches off all debugging commands.

debug=true

This setting is provided by the glossaries package and switches on logging information if an entry is indexed before the relevant indexing files have been opened (only applicable with `makeindex` and `xindy`). This option is extended by glossaries-extra to also display the label of unknown entries before the `??` marker.

```
\documentclass{article}
\usepackage[debug]
 {glossaries-extra}
\begin{document}
\gls{example}
\end{document}
```

```
[example]??
```

This uses `\glsshowtargetfonttext` for the annotation, which is provided by the base glossaries package.

debug=showaccsupp

Provided by glossaries, this setting shows accessibility information (glossaries-accsupp).

debug=all

Implements all debugging options.

debug=showwrgloss

This setting is only available with glossaries−extra. This implements debug=true and shows a marker (·) just before the write operation is performed by indexing commands. With record=hybrid there will be two marks: one for the write operation to the aux file and one for the associated glossary file used by makeindex/xindy. The marker is produced with the command:

```
\glsxtrwrglossmark
```

If the indexcounter option has been used, this setting will also mark where the wrglossary counter has been incremented. The marker is produced with the command:

```
\glsxtrwrglosscountermark{⟨number⟩}
```

This marker is also inserted before the location in the definition of \glsxtrwrglossarylocfmt.

debug=showtargets

This setting is provided by glossaries and displays the hyperlink target names whenever any glossary-related commands create a hyperlink or hypertarget (for example, \gls, \glslink, \glstarget or \glshypernumber). The default is to use marginal notes in TEX's "outer" mode and inline annotations for "inner" or maths modes. This uses \glsshowtargetinner for inner and maths annotations and \glsshowtargetouter for the outer annotation.

If there are many targets within a single paragraph this can lead to "too many floats", so glossaries−extra provides a new package option showtargets that can be used to easily switch to inline annotations for outer mode (rather than having to redefine \glsshowtargetouter).

showtargets=⟨*value*⟩

Automatically implements debug=showtargets and adjusts the annotations according to the ⟨*value*⟩. The glossaries−extra package provides supplementary commands to support this option.

```
\glsxtrshowtargetouter{⟨target-name⟩}
```

Formats annotations in outer mode. This is initially \glsshowtargetouter to match debug=showtargets.

\glsxtrshowtargetinner{⟨target-name⟩}

Formats annotations in inner mode. This is initially \glsshowtargetinner to match debug=showtargets.

\glsshowtargetinnersymleft{name}

Shows the left annotation and marker. This uses the left symbol marker:

\glsxtrshowtargetsymbolleft

\glsshowtargetinnersymright{name}

Shows the right marker and annotation. This uses the right symbol marker:

\glsxtrshowtargetsymbolright

showtargets=left

A marker is placed to the left of the link/target and a marginal note is used in outer mode.

showtargets=right

A marker is placed to the right of the link/target and a marginal note is used in outer mode.

showtargets=innerleft

A marker and annotation are placed to the left of the link/target in all modes.

showtargets=innerright

A marker and annotation are placed to the right of the link/target in all modes.

showtargets=annoteleft

Markers are placed on either side of the link/target with the annotation on the left in all modes.

```
showtargets=annoteright
```

Markers are placed on either side of the link/target with the annotation on the right in all modes.

3. Defining Entries

The base glossaries package provides commands, such as \newglossaryentry, to define entries. The glossaries-extra package provides some additional commands, described in §3.1. For abbreviations, see §4. If you use bib2gls, it will write command definitions within the glstex file. See the bib2gls user manual for further information about those commands.

The glossaries user manual warns against using commands such as \gls within field values. However, if you really need this, the glossaries-extra package provides \glsxtrp (see §5.4). Alternatively, you may want to consider multi (compound) entries instead (see §7).

3.1. Command Definitions

\longnewglossaryentry{⟨*entry-label*⟩}{⟨*key=value list*⟩}{⟨*description*⟩}

This command is provided by the base glossaries package to cater for entries with descriptions that contain paragraph breaks. (The ⟨*key*⟩=⟨*value*⟩ interface doesn't support paragraph breaks in the value.) The base package only provides an unstarred version of this command, which automatically inserts \leavevmode\unskip\nopostdesc at the end of the description. The glossaries-extra package replaces this with a single command:

\glsxtrpostlongdescription

which has the same effect, but can be redefined if required.

The glossaries-extra package provides a starred form:

\longnewglossaryentry*{⟨*entry-label*⟩}{⟨*key=value list*⟩}{⟨*description*⟩}

This doesn't insert the hook at the end of the description.

For a general purpose post-description hook, see §8.6.2.

Additionally, the symbols package option provides \glsxtrnewsymbol, and the numbers package option provides \glsxtrnewnumber. See §2.1 for further details.

3.2. Glossary Entry Keys

In addition to the glossary entry keys provided by the base glossaries package (summarised in §II) the glossaries−extra package provides:

category=⟨*category-label*⟩ *initial:* **general**

Assigns the category label. This should not contain any special or active characters as it's used to form command names. See §10 for further details.

seealso={⟨*xr-list*⟩}

This key is analogous to the `see` key but the tag is always given by `\seealsoname`. The value ⟨*xr-list*⟩ should be a comma-separated list of entry labels. As with the `see` key, this key automatically indexes the cross-reference by default. The cross-reference will be displayed in the location list using `\glsxtruseseealsoformat` (see §5.9). Use `autoseeindex=false` to prevent the automatic indexing. (With `bib2gls`, adjust the `selection` criteria.)

With just the base glossaries package, the `see` key simply performs this automated indexing. With glossaries−extra the value is also saved. Similarly with the `seealso` key. The value isn't saved with explicit use of `\glsxtrindexseealso` or `\glssee`.

alias={⟨*xr-label*⟩}

This is similar to the `see` key but the value can only be a single entry label. In addition to automatically indexing the cross-reference, this command will cause the entry with this key to have hyperlinks go to the aliased entry when referenced with commands like `\gls`. Whenever the entry is indexed with commands like `\gls`, the indexing will be performed on the target entry (the `alias` value). See §5.9 for further details.

Any entry that has a `see`, `seealso` or `alias` key set will be added to the glossary by default when using `makeindex` or `xindy`. If you don't want this behaviour, use the `autoseeindex=false` package option and implement a post-description hook to insert the cross-reference. Alternatively, consider switching to `bib2gls`.

If you use `bib2gls` (see §11) then most of the glossary entry keys can be used as analogous fields in the `bib` file. For example, instead of writing the following code in your `tex` file:

```
\newglossaryentry{duck}{name={duck},
 description={a waterbird with webbed feet}}
\newabbreviation{svm}{SVM}{support vector machine}
```

You would write the following in a bib file:

```
@entry{duck,
 name={duck},
 description={a waterbird with webbed feet}
}
@abbreviation{svm,
  short={SVM},
  long={support vector machine},
}
```

There are, however, some keys that are considered internal fields by bib2gls, in that they are defined as keys by glossaries-extra and may be assigned in the glstex file that's input by \GlsXtrLoadResources, but they should not be used in the bib files.

For example, the sort key (which is recommended with xindy where the name contains symbols) should not be used in the bib file. Instead, use the sort-field resource option or the system of sort fallbacks to choose the most appropriate field to obtain the sort value (see Gallery: Sorting[1]). The group and location keys are also considered internal fields and are only applicable with the "unsrt" family of commands.

The group and location keys are defined by the record=only and record=nameref options and are only applicable with the "unsrt" family of commands.

group={⟨*group-label*⟩}

This key is used by bib2gls within the glstex file to set the group label. This label is typically a by-product of the sorting method (see §8.6.4). If it is explicitly set without reference to the order it can result in fragmented groups, see Gallery: Logical Glossary Divisions (type vs group vs parent).[2] The group title can be set with \glsxtrsetgrouptitle. You will need to invoke bib2gls with the --group (or -g) switch to ensure that this key is set, when required.

[1] dickimaw-books.com/gallery/index.php?label=bib2gls-sorting
[2] dickimaw-books.com/gallery/index.php?label=logicaldivisions

Letter groups are a consequence of sorting, not the other way around.

location={⟨*location-list*⟩}

Used by bib2gls to store the formatted location list. The unformatted internal location list is stored in the loclist field, as with \printnoidxglossary.

With the "unsrt" family of commands, if the location field isn't set, then it will try the loclist field instead, using the same method as \printnoidxglossary to display the locations. If you don't want locations with bib2gls, either use nonumberlist or use the save-locations=false resource option.

The base glossaries package provides \glsaddkey and \glsaddstoragekey to allow custom keys to be defined. The glossaries-extra package additionally provides:

\glsxtrprovidestoragekey{⟨*key*⟩}{⟨*default value*⟩}{⟨*no link cs*⟩}
*modifier: **

This is like \glsaddstoragekey but does nothing if the key has already been defined. As with \glsaddstoragekey, the starred version switches on field expansion for the given key (provided that it hasn't already been defined).

\glsxtrifkeydefined{⟨*key*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if the given key has been defined as a glossary entry key.

3.3. Plurals

Some languages, such as English, have a general rule that plurals are formed from the singular with a suffix appended. This isn't an absolute rule. There are plenty of exceptions (for example, geese, children, churches, elves, fairies, sheep). The glossaries package allows the plural key to be optional when defining entries. In some cases a plural may not make any sense (for example, the term is a symbol) and in some cases the plural may be identical to the singular.

To make life easier for languages where the majority of plurals can simply be formed by appending a suffix to the singular, the glossaries package lets the plural field default to the value of the text field with \glspluralsuffix appended. This command is defined to be just the letter "s". This means that the majority of terms don't need to have the plural supplied as well, and you only need to use it for the exceptions.

For languages that don't have this general rule, the plural field will always need to be supplied, where needed.

There are other plural fields, such as firstplural, longplural and shortplural. Again, if you are using a language that doesn't have a simple suffix rule, you'll have to supply the plural forms if you need them (and if a plural makes sense in the context).

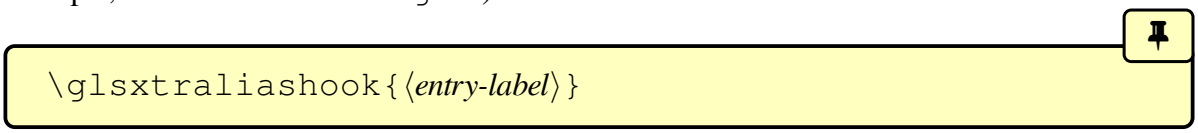
If these fields are omitted, the glossaries package follows these rules:

- If firstplural is missing, then \glspluralsuffix is appended to the first field, if that field has been supplied. If the first field hasn't been supplied but the plural field has been supplied, then the firstplural field defaults to the plural field. If the plural field hasn't been supplied, then both the plural and firstplural fields default to the text field (or name, if no text field) with \glspluralsuffix appended.
- If the longplural field is missing, then \glspluralsuffix is appended to the long field, if the long field has been supplied.
- If the shortplural field is missing then, *with the base glossaries acronym mechanism*, \acrpluralsuffix is appended to the short field.

The *last case is changed* with glossaries-extra. With this extension package, the shortplural field defaults to the short field with \abbrvpluralsuffix appended unless overridden by category attributes. This suffix command is set by the abbreviation styles. This means that every time an abbreviation style is implemented, \abbrvpluralsuffix is redefined, see §4.1.2 for further details.

3.4. Entry Aliases

An entry can be made an alias of another entry using the alias key. The value should be the label of the other term. There's no check for the other's existence when the aliased entry is defined. This is to allow the possibility of defining the other entry after the aliased entry. (For example, when used with bib2gls.)



\glsxtraliashook{⟨*entry-label*⟩}

This hook is implemented when an entry is defined with the alias key set. It does nothing by default. The value of the alias field can be obtained with \glsxtralias{⟨*entry-label*⟩}.

If an entry ⟨*entry-1*⟩ is made an alias of ⟨*entry-2*⟩ then:

- If the see field wasn't provided when ⟨*entry-1*⟩ was defined, the alias key will automatically trigger

  \glssee{⟨*entry-1*⟩}{⟨*entry-2*⟩}

- If the hyperref package has been loaded then `\gls{`⟨*entry-1*⟩`}` will link to ⟨*entry-2*⟩'s target. (Unless the `targeturl` attribute has been set for ⟨*entry-1*⟩'s category.)
- With `record=off` or `record=hybrid`, the `noindex` setting will automatically be triggered when referencing ⟨*entry-1*⟩ with commands like `\gls` or `\glstext`. This prevents ⟨*entry-1*⟩ from having a location list (aside from the cross-reference added with `\glssee`) unless it's been explicitly indexed with `\glsadd` or if the indexing has been explicitly set using `noindex=false`. See §5.9.3 for adjusting the indexing hook.

  Note that with `record=only`, the location list for aliased entries is controlled with `bib2gls`'s settings.

The value of the `alias` field can be accessed with `\glsxtralias` (see §5.9.2).

3.5. Setting or Updating Fields

See §5.11 for accessing field values and §5.15 for testing field values.

Modifications to fields only have an effect from that point onwards and may be localised to the current scope. If you are using `docdef=true`, any changes to the field values won't be saved in the `glsdefs` file.

Some of these commands are subtly different from each other. For example, `\glsfielddef` (provided by the base glossaries package), `\glsxtrdeffield` and `\GlsXtrSetField` all assign a value to a field, but `\glsfielddef` requires that both the entry and the field exists (so it can't be used to set an unknown internal field), `\GlsXtrSetField` requires that the entry exists (so it can be used to set an internal field that doesn't have an associated key provided that the entry has been defined), and `\glsxtrdeffield` doesn't perform any existence checks (which means that it can be used to assign internal fields before the entry is actually defined).

The commands described in this section don't require the field to have an associated glossary entry key, so you need to be careful not to misspell the field labels.

Assigning or changing fields using the commands described here won't alter related fields. For example, if you use the `text` key but not the `plural` key when you define an entry with `\newglossaryentry`, the `plural` key will automatically be set as well, but if you change the value of the `text` field after the entry has been defined, the `plural` field won't be changed. Particular care is required if the field contributes in some way to the indexing information, as this information is typically initialised when the entry is first defined. This includes the `sort` and `parent` keys, which should not be changed after the entry has been defined.

With bib2gls, entries aren't defined on the first LaTeX run. This means that commands that test for existence will produce a warning and (within the document environment) the ?? unknown marker. For example:

```
\documentclass{article}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[src={myentries},selection=all]
\begin{document}
Defining info
\glsxtrdeffield{sample}{info}{some information}.
Defining note
\GlsXtrSetField{sample}{note}{some note}.

Info: \glsxtrusefield{sample}{info}.
Note: \glsxtrusefield{sample}{note}.
\end{document}
```

On the first LaTeX run, this produces:

Defining info . Defining note ??.
Info: some information. Note: .

At this point the sample entry hasn't been defined, so referencing it in \GlsXtrSetField results in a warning and the double question mark ?? unknown marker in the text. The field (note) isn't saved, so nothing is shown when the field is referenced with \glsxtrusefield. Whereas \glsxtrdeffield does save the field with the label info associated with the label sample, even though the sample entry hasn't actually been defined. The field can then be later obtained with \glsxtrusefield. Once bib2gls has been run, the sample entry should now have its definition in the glstex file, which is loaded by \GlsXtrLoadResources and the note field can be set.

\glsxtrdeffield{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}

This uses etoolbox's \csdef command to locally set the field given by its internal label ⟨*field-label*⟩ to ⟨*value*⟩ for the entry identified by ⟨*entry-label*⟩. No existence check is performed.

\glsxtredeffield{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}

This is like \glsxtrdeffield but (protected) fully expands the value before assigning it to the field.

\glsxtrapptocsvfield{⟨entry-label⟩}{⟨field-label⟩}{⟨element⟩}

This command is designed for fields that should contain comma-separated lists. If the field hasn't been defined, this behaves like \glsxtrdeffield otherwise it will append a comma followed by ⟨element⟩ (unexpanded) to the field value. No existence check is performed. This field can be iterated over using \glsxtrforcsvfield or formatted using \glsxtrfieldformatcsvlist. See §5.13 for further details.

\glsxtrfieldlistadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}

Appends the given value to the given entry's field (identified using the field's internal label) using etoolbox's \listcsadd. The field value can later be iterated over using \glsxtrfielddolistloop or \glsxtrfieldforlistloop.

\glsxtrfieldlistgadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}

As above but uses \listcsgadd to make a global change.

\glsxtrfieldlisteadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}

As above but uses \listcseadd to expand the value.

\glsxtrfieldlistxadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}

As above but uses \listcsxadd to make a global change.

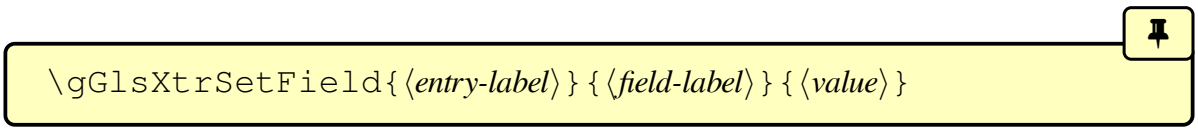
\glsxtrsetfieldifexists{⟨entry-label⟩}{⟨field-label⟩}{⟨code⟩}

This is used by the commands \GlsXtrSetField, \gGlsXtrSetField, \xGlsXtrSetField, \eGlsXtrSetField, \GlsXtrLetField, \csGlsXtrLetField and \GlsXtrLetFieldToField to produce an error (or warning with undefaction=warn) if the entry doesn't exist. This can be redefined to add extra checks (for example, to prohibit changing certain fields).

\GlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}

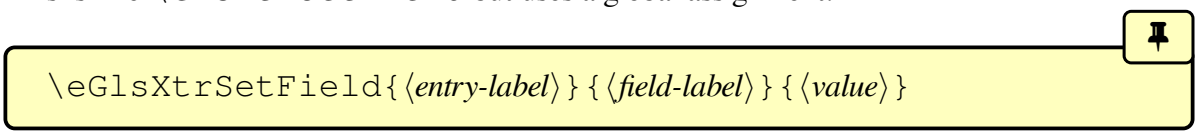
This uses etoolbox's \csdef command to locally set the field given by its internal label ⟨field-label⟩ to ⟨value⟩ for the entry identified by ⟨entry-label⟩.

This command is written to the glstex file by bib2gls to set fields that don't have a corresponding key.



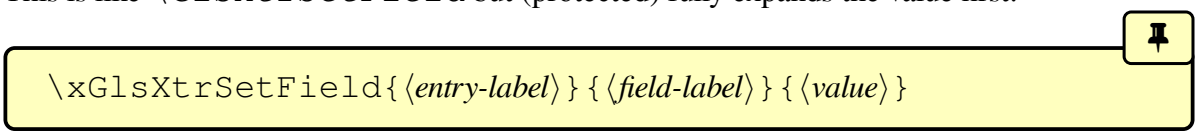
\gGlsXtrSetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}

This is like \GlsXtrSetField but uses a global assignment.



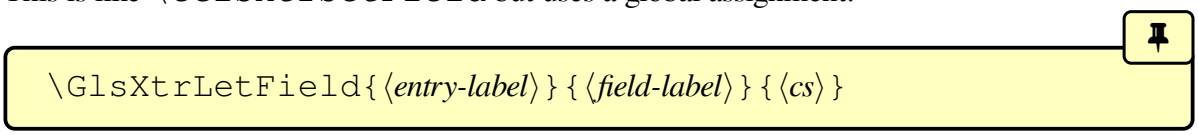
\eGlsXtrSetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}

This is like \GlsXtrSetField but (protected) fully expands the value first.



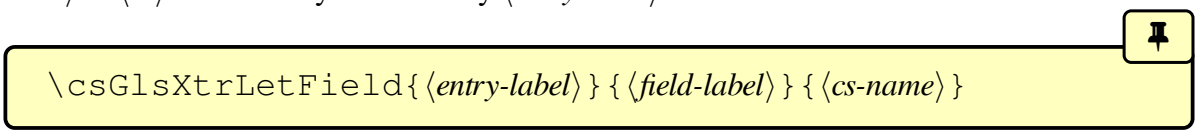
\xGlsXtrSetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}

This is like \eGlsXtrSetField but uses a global assignment.



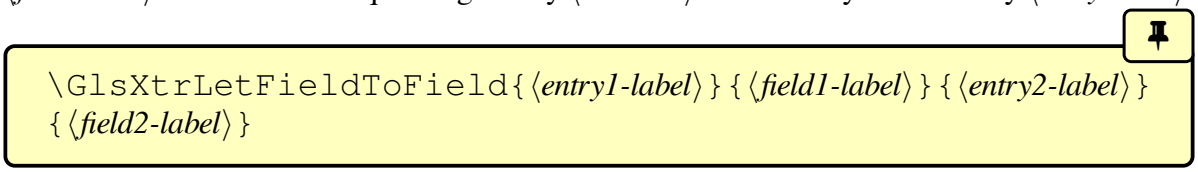
\GlsXtrLetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*cs*⟩}

This uses etoolbox's \cslet command to locally set the field given by its internal label ⟨*field-label*⟩ to ⟨*cs*⟩ for the entry identified by ⟨*entry-label*⟩.



\csGlsXtrLetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*cs-name*⟩}

This uses etoolbox's \csletcs command to locally set the field given by its internal label ⟨*field-label*⟩ to the control sequence given by ⟨*cs-name*⟩ for the entry identified by ⟨*entry-label*⟩.



\GlsXtrLetFieldToField{⟨*entry1-label*⟩}{⟨*field1-label*⟩}{⟨*entry2-label*⟩}{⟨*field2-label*⟩}

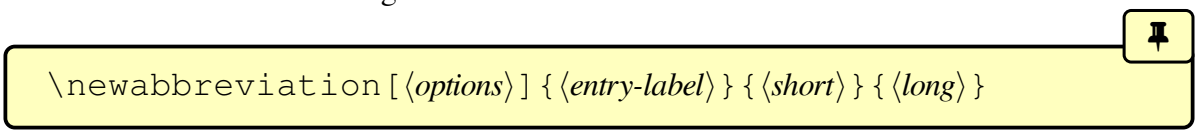
This assigns the field identified by its internal label ⟨*field1-label*⟩ for the entry identified by ⟨*entry1-label*⟩ to the value of the field identified by ⟨*field2-label*⟩ for the entry identified by ⟨*entry2-label*⟩

4. Abbreviations

The acronym mechanism implemented by the base glossaries package is insufficiently flexible for some documents. The glossaries−extra package provides a completely different mechanism to deal with abbreviations in a more flexible manner. The two methods are incompatible. However, the glossaries−extra package provides predefined styles that emulate the appearance of the styles provided by the base package. If you have previously used just the base glossaries package, consult Table 4.2 for the closest matching abbreviation style.

4.1. Defining Abbreviations

Abbreviations are defined using:



```
\newabbreviation[⟨options⟩]{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}
```

where ⟨*entry-label*⟩ is the entry's label (used in commands like `\gls`), ⟨*short*⟩ is the short form (the abbreviation) and ⟨*long*⟩ is the long form (what the abbreviation is short for). The optional argument ⟨*options*⟩ may be used to set additional keys (as per the options list in `\newglossaryentry`), such as `type` or `category`.

This command internally uses `\newglossaryentry` and sets the `type` to `\glsxtrabbrvtype` and the `category` to `abbreviation`. The category (see §10) determines the abbreviation style. The style for a particular category is set using `\setabbreviationstyle`. If the optional argument is omitted, the `abbreviation` category is assumed (see §4.5 for further details).

The following example document sets up three different abbreviation styles: long−short−sc for the `abbreviation` category, long−only−short−only for the custom `genus` category, and short−nolong for the custom `common` category. Note that the custom `title` category doesn't have an associated style.

```
\setabbreviationstyle{long-short-sc}
\setabbreviationstyle[genus]{long-only-short-only}
\setabbreviationstyle[common]{short-nolong}
\newabbreviation{xml}{xml}
{extensible markup language}
\newabbreviation[category={genus}]{clostridium}{C.}
{Clostridium}
```

```
\newabbreviation[category={genus}]{myristica}{M.}
{Myristica}
\newabbreviation[category={common}]{html}{HTML}
{hypertext markup language}
\newabbreviation[category={title}]{dr}{Dr}{Doctor}
\begin{document}
First use: \gls{xml}, \gls{clostridium},
\gls{myristica}, \gls{html}, \gls{dr}.

Next use: \gls{xml}, \gls{clostridium},
\gls{myristica}, \gls{html}, \gls{dr}.
\end{document}
```

Example 1: Multiple abbreviation styles

First use: extensible markup language (XML), Clostridium, Myristica, HTML, Doctor (DR).
Next use: XML, C., M., HTML, DR.

If the category doesn't have an associated style, the style for the `abbreviation` category will be used, as with the `dr` entry above, which uses the long-short-sc style because no style has been associated with its custom `title` category.

There are two categories that have an abbreviation style set by default: `abbreviation` and `acronym`. These are initialised as follows:

```
\setabbreviationstyle{long-short}
\setabbreviationstyle[acronym]{short}
```

This means that abbreviations defined with the default `abbreviation` category will show the long form followed by the short form in parentheses on first use, and those defined with the `category` set to `acronym` will only show the short form (that is, the long form won't be shown on first use).

To make it easier to migrate a file containing entries defined with `\newacronym`, the glossaries-extra package redefines `\newacronym` to do:

```
\newabbreviation
 [type=\acronymtype,category=acronym,⟨options⟩]
 {⟨entry-label⟩}{⟨short⟩}{⟨long⟩}
```

Note that this sets the `category` to `acronym`, which means that any abbreviations defined

with \newacronym will use the short style by default. If you want to use a different style, you need to set the abbreviation style for the acronym category. For example, to use the long–short style:

```
\setabbreviationstyle[acronym]{long-short}
```

This must be placed before the first instance of \newacronym.

You can't use \setacronymstyle with glossaries–extra.

If you have defined any acronym styles with \newacronymstyle, you will have to migrate them over to \newabbreviationstyle. However, most of the predefined abbreviation styles are flexible enough to adapt to common abbreviation formats. It is possible to revert \newacronym back to using the base glossaries package's acronym mechanism (§4.6), but it should generally not be necessary.

Terms defined with \newabbreviation (and \newacronym) can be referenced in the main document text using commands like \gls. (If you want to use shortcut commands like \ac, use the shortcuts=ac package option.) Remember that you can use the preset and preunset options to reset or unset the first use flag (see §5.10). Alternatively, you can use the commands described in §4.3. For headings and captions, see §5.3.2.

Avoid using \glsfirst, \glsfirstplural, \glstext and \glsplural with abbreviations. Many of the abbreviation styles are too complex to work with these commands (particularly the case-changing variants or with the ⟨*insert*⟩ final optional argument or with innertextformat). Instead, use commands like \gls, \glsxtrshort, \glsxtrlong and \glsxtrfull.

4.1.1. Abbreviation Fields: long and short

The ⟨*short*⟩ and ⟨*long*⟩ arguments are internally assigned with the short and long keys (so don't use those keys in ⟨*options*⟩), but the short and long values may first be modified by category attributes, such as markwords or markshortwords. As with other entries, avoid nested links (see §5.4). This means avoid using the \gls-like and \glstext-like commands within ⟨*short*⟩ and ⟨*long*⟩.

If an abbreviation can be formed by combining other entries, consider using the multi (compound) entry function (see §7).

4.1.2. Abbreviation Fields: longplural and shortplural

The longplural key defaults to ⟨*long*⟩\glspluralsuffix and the shortplural key defaults to ⟨*short*⟩\abbrvpluralsuffix. The aposplural attribute will instead set the shortplural to ⟨*short*⟩'\abbrvpluralsuffix and the noshortplural attribute will set shortplural to just ⟨*short*⟩ (see §10). If these values are not appropriate, you will need to explicitly set the longplural and shortplural keys in ⟨*options*⟩.

The short plural suffix \abbrvpluralsuffix is redefined by the abbreviation style. Some styles, such as the long-short style, simply redefine \abbrvpluralsuffix to just:

\glsxtrabbrvpluralsuffix *initial:* \glspluralsuffix

which is defined to \glspluralsuffix.

Some styles, such as the long-short-sc style, redefine \abbrvpluralsuffix to include code to counteract the formatting of the abbreviation.

If you want to change the default short plural suffix, you need to redefine \glsxtrabbrvpluralsuffix not \abbrvpluralsuffix. If you don't want the suffix added, then set the noshortplural attribute to true.

4.1.3. Abbreviation Fields: name and description

The name key is set according to the abbreviation style. There should not be any need to explicitly set it. Some styles require the description key to be set in ⟨*options*⟩, but other styles will set the description to the long form.

4.1.4. Abbreviation Fields: type

Abbreviations can be assigned to a particular glossary using the type key in ⟨*options*⟩. The default for \newabbreviation is:

\glsxtrabbrvtype *initial:* \glsdefaulttype

This is initialised to \glsdefaulttype (the default glossary), but the abbreviations package option will redefine it to abbreviations.

The default type for \newacronym is:

\acronymtype *initial:* \glsdefaulttype

This is initialised to \glsdefaulttype, but the acronyms package option will redefine it to acronym.

4.1.5. General Hooks

The following are general purpose hooks used within \newabbreviation. Note that there are additional hooks that are used by the abbreviation styles (see §4.5.3.1).

\glsxtrnewabbrevpresetkeyhook{⟨options⟩}{⟨label⟩}{⟨short⟩}

This hook is provided for further customisation, if required. It's implemented before the entry is defined (before the shortplural and longplural keys supplied in ⟨options⟩ are parsed). Does nothing by default. The arguments are a legacy throwback to old versions that didn't have \glsxtrorgshort.

\newabbreviationhook

This hook is performed just before the entry is defined. Does nothing by default.

4.2. Examples: makeindex vs bib2gls

Example 2 is a simple document that uses makeindex:

2

```
\documentclass{article}
\usepackage{glossaries-extra}
\makeglossaries
\newglossaryentry{sample}{name={sample},
 description={an example}}
\newabbreviation{xml}{XML}
{extensible markup language}
\newacronym{nasa}{NASA}
{National Aeronautics and Space Administration}
\begin{document}
First use: \gls{sample}, \gls{xml} and \gls{nasa}.
Next use: \gls{sample}, \gls{xml} and \gls{nasa}.
\printglossaries
\end{document}
```

Example 2: \newabbreviation vs \newacronym vs \newglossaryentry

First use: sample, extensible markup language (XML) and NASA. Next use: sample, XML and NASA.

Glossary

NASA National Aeronautics and Space Administration 1

sample an example 1

XML extensible markup language 1

Note that the long form of NASA isn't displayed on the first use of \gls{nasa}. This is because the acronym category uses the short style by default.

In the above example, all entries are placed in the main (default) glossary. The package options abbreviations and acronyms can be used to split them off into separate glossaries.

If you use bib2gls, the analogous bib entry types are @abbreviation and @acronym. Example 3 modifies the above Example 2 to use bib2gls:

```
\documentclass{article}
\begin{filecontents*}{\jobname.bib}
@entry{sample,
 name={sample},
 description={an example}
}
@abbreviation{xml,
 short={XML},
 long={extensible markup language}
}
@acronym{nasa,
 short={NASA},
 long={National Aeronautics and
   Space Administration}
}
\end{filecontents*}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources
\begin{document}
First use: \gls{sample}, \gls{xml} and \gls{nasa}.
```

```
Next use: \gls{sample}, \gls{xml} and \gls{nasa}.
\printunsrtglossaries
\end{document}
```

Example 3: `@abbreviation` vs `@acronym` vs `@entry`

First use: sample, extensible markup language (XML) and NASA. Next use: sample, XML and NASA.

Glossary

NASA National Aeronautics and Space Administration 1

sample an example 1

XML extensible markup language 1

4.3. Referencing (Using) Abbreviations

Since `\newabbreviation` internally uses `\newglossaryentry`, you can reference abbreviations with the `\gls`-like commands as with other entries. Remember that you can use the `prereset` and `preunset` options to reset or unset the first use flag (see §5.10).

In general it's best not to use `\glsfirst`, `\glsfirstplural`, `\glstext`, `\glsplural` or their case-changing variants as many of the abbreviation styles are too complicated for those commands. If you specifically want the full form, use `\gls` with `prereset` or use `\glsxtrfull`. If you specifically want the short form for a particular instance, use `\gls` with `preunset` or use `\glsxtrshort`. If you only want the long form for a particular instance, use `\glsxtrlong`.

If you never want the short form with `\gls`, use one of the "noshort" styles, such as long−noshort. If you never want the long form with `\gls`, use one of the "nolong" styles, such as short−nolong.

If you need to use abbreviations in headings or captions, use commands like `\glsfmtshort` and `\glsfmtlong` (see §5.3.2). Commands like `\glsentryname` are likely to contain non-expandable content.

Example 4 defines abbreviations and references them in a document with hyperlink support:

```
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\newabbreviation{svm}{SVM}{support vector machine}
\newabbreviation{html}{HTML}
{hypertext markup language}

\begin{document}
\tableofcontents
\section{Introducing the \glsfmtlong{svm}}
First use: \gls{svm}.
Next use: \gls{svm}.

\section{Introducing \gls{html} (incorrect)}
First use (not!): \gls{html}.
Next use: \gls{html}.

\glsreset{html}
\section{Introducing \glsxtrshort{html} (incorrect)}
First use: \gls{html}.
Next use: \gls{html}.

\glsreset{html}
\section{Introducing \glsfmtshort{html}}
First use: \gls{html}.
Next use: \gls{html}.

\printglossaries
\end{document}
```

In Example 4, compare the first section heading (which references an abbreviation with `\glsfmtlong`) with the second section heading (which references an abbreviation with `\gls`). Note that the first use of the `html` entry actually occurs in the table of contents, which results in the full form showing in the table of contents, but only the abbreviation is shown in the actual section 2 heading. The PDF bookmark shows the entry label (`html`) not the abbreviation (HTML). There is also a nested link for section 2 in the table of contents. In some PDF viewers (such as Okular), this will lead to section 2 but, in others (such as Evince), it will lead to the HTML entry target in the glossary. Similarly for section 3.

As with the base glossaries package, the unformatted short and long forms can be obtained with `\glsentryshort` and `\glsentrylong` or, for the plural forms, `\glsentryshortpl` and `\glsentrylongpl`. These are analogous to commands like `\glsentrytext` and may be used in expandable contexts. The sentence case versions (`\Glsen-`

Example 4: Referencing an abbreviation (with hyperref)

Contents

1 Introducing the support vector machine

First use: support vector machine (SVM). Next use: SVM.

2 Introducing HTML (incorrect)

First use (not!): HTML. Next use: HTML.

3 Introducing HTML (incorrect)

First use: hypertext markup language (HTML). Next use: HTML.

4 Introducing HTML

First use: hypertext markup language (HTML). Next use: HTML.

Glossary

HTML hypertext markup language 1

SVM support vector machine 1

tryshort, \Glsentrylong, \Glsentryshort, and \Glsentrylong) are all robust in glossaries v4.49 and lower. As from glossaries v4.50, they can expand in PDF bookmarks, but outside of PDF bookmarks they will expand to a robust internal command.

Don't use the base glossaries package's acronym commands, such as \acrshort. These aren't compatible with \newabbreviation.

Each abbreviation style has a display full form, which is the format produced with the first use of \gls, and an inline full form, which is the format produced by \glsxtrfull. For some styles, such as long–short, the display and inline forms are identical.

Example 5 demonstrates the difference between the first use of \gls compared with the inline full form for the footnote abbreviation style. The example also uses \glsfirst to demonstrate that it produces an undesirable result with the selected abbreviation style.

```
\setabbreviationstyle{footnote}
\newabbreviation{nasa}{NASA}
{National Aeronautics and Space Administration}

\begin{document}
\gls{nasa}['s] space exploration\ldots

\glsxtrfull{nasa}['s] space exploration\ldots

\glsfirst{nasa}['s] space exploration\ldots
\end{document}
```

In Example 5, the first use of \gls puts the long form in the footnote but correctly inserts the final optional argument before the footnote marker. The inline full form (obtained with \glsxtrfull) doesn't use a footnote, but instead shows the long form in parentheses after the short form. The insert material is correctly placed after the short form. Compare this with the final line, which uses \glsfirst. This shows the long form in the footnote, but the inserted material can't be shifted before the footnote marker, which results in the strange "NASA[2]'s".

The following commands are included in the set of \glstext-like commands. They have the same options as \glstext and don't change the first use flag. They will index (unless noindex is used), create a hyperlink (if enabled), and use the post-link hook.

\glsxtrshort [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the short form using the abbreviation style's formatting.

Example 5: First use of `\gls` vs `\glsxtrfull` vs `\glsfirst`

NASA's[1] space exploration...

NASA's (National Aeronautics and Space Administration) space exploration...

NASA[2]'s space exploration...

[1]National Aeronautics and Space Administration

[2]National Aeronautics and Space Administration

\Glsxtrshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but sentence case version.

\GLSxtrshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but all caps version.

\glsxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the short plural form using the abbreviation style's formatting.

\Glsxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but sentence case version.

\GLSxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but all caps version.

\glsxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the long form using the abbreviation style's formatting. As from v1.49, this command simulates first use by defining `\glsxtrifwasfirstuse` to do its first argument. This is done via the command:

\glsxtrsetlongfirstuse{⟨*entry-label*⟩}

which is defined as:

```
\newcommand{\glsxtrsetlongfirstuse}[1]{%
 \let\glsxtrifwasfirstuse\@firstoftwo
}
```

This command takes the entry label as the argument, which is ignored by default.

To restore the original behaviour, redefine this command as follows:

```
\renewcommand{\glsxtrsetlongfirstuse}[1]{%
 \letcs\glsxtrifwasfirstuse{@secondoftwo}%
}
```

This command is also used by the case-changing and plural variants listed below.

`\Glsxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but sentence case version.

`\GLSxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but all caps version.

`\glsxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

Displays the long plural form using the abbreviation style's formatting.

`\Glsxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but sentence case version.

`\GLSxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but all caps version.

`\glsxtrfull[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

Displays the inline full form using the abbreviation style's formatting. Depending on the style, this may not be the same as the text produced with the first use of `\gls`.

`\Glsxtrfull[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but sentence case version.

`\GLSxtrfull[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]` *modifiers: * + ⟨alt-mod⟩*

As above, but all caps version.

\glsxtrfullpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the inline full plural form using the abbreviation style's formatting. Depending on the style, this may not be the same as the text produced with the first use of \glspl.

\Glsxtrfullpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but sentence case version.

\GLSxtrfullpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As above, but all caps version.

\glsxtrsetupfulldefs

This hook is used within \glsxtrfull, \glsxtrfullpl and the case-changing variations to initialise \glsxtrifwasfirstuse in case it's required in the post-link hook. The default definition is to simulate first use. Note that changing this can cause unexpected results with abbreviation styles that set the post-link hook, such as short−postlong−user.

\glsxtrfullsaveinsert{⟨entry-label⟩}{⟨insert⟩}

This hook is used within \glsxtrfull, \glsxtrfullpl and the case-changing variations to initialise the \glsinsert placeholder. The default definition is to use \glsxtrsaveinsert. If the insert isn't saved, it can't be used within the post-link hook for the \glsxtrfull etc. This affects the behaviour of the "post-hyphen" abbreviation styles, such as long−hyphen−postshort−hyphen.

4.3.1. Prefixes

If you are using the glossaries−prefix package (which can be loaded via the prefix package option), then there are commands similar to \glsxtrshort and \glsxtrlong that insert the corresponding prefix and separator at the front if the short or long form, if the prefix has been set and is non-empty. In all cases, the separator is \glsprefixsep, as with \pgls.

These commands require glossaries−prefix.

\pglsxtrshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \glsxtrshort but inserts the prefix field and separator, if the prefix value is set and non-empty.

\Pglsxtrshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrshort but sentence case. Note the initial "P" in the command name, which matches \Pgls (similarly for the other prefix sentence case commands).

\PGLSxtrshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrshort but all caps.

\pglsxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \glsxtrshortpl but inserts the prefixplural field and separator, if the prefixplural value is set and non-empty.

\Pglsxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrshortpl but sentence case.

\PGLSxtrshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrshortpl but all caps.

\pglsxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \glsxtrlong but inserts the prefixfirst field and separator, if the prefixfirst value is set and non-empty.

\Pglsxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrlong but sentence case.

\PGLSxtrlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrlong but all caps.

\pglsxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \glsxtrlongpl but inserts the prefixfirstplural field and separator, if the prefixfirstplural value is set and non-empty.

\Pglsxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrlongpl but sentence case.

\PGLSxtrlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsxtrlongpl but all caps.

4.3.2. Abbreviation Shortcut Commands

The abbreviation shortcut commands can be enabled using the shortcuts=abbreviations package option (or shortcuts=abbr or shortcuts=ac). The provided shortcut commands are listed in Table 4.1. Note that \glsxtrenablerecordcount will switch the shortcuts that use the \cgls-like commands to the corresponding \rgls-like command.

4.4. Tagging Initials

Initial tagging allows you to highlight the initials that form the abbreviation when the long form is shown in the glossary.

\GlsXtrEnableInitialTagging{⟨*categories*⟩}{⟨*cs*⟩} *modifier:* *

\GlsXtrEnableInitialTagging must be placed before the abbreviations are defined.

This command (robustly) defines ⟨*cs*⟩ (a control sequence) to accept a single argument, which is the letter (or letters) that needs to be tagged. The normal behaviour of ⟨*cs*⟩ within the document

Table 4.1.: Abbreviation Shortcut Commands

Shortcut (shortcuts=abbreviations)	**Shortcut (shortcuts=ac)**	**Equivalent Command**
\ab	\ac	\cgls
\abp	\acp	\cglspl
\as	\acs	\glsxtrshort
\asp	\acsp	\glsxtrshortpl
\al	\acl	\glsxtrlong
\alp	\aclp	\glsxtrlongpl
\af	\acf	\glsxtrfull
\afp	\acfp	\glsxtrfullpl
\Ab	\Ac	\cgls
\Abp	\Acp	\cglspl
\As	\Acs	\Glsxtrshort
\Asp	\Acsp	\Glsxtrshortpl
\Al	\Acl	\Glsxtrlong
\Alp	\Aclp	\Glsxtrlongpl
\Af	\Acf	\Glsxtrfull
\Afp	\Acfp	\Glsxtrfullpl
\AB	\AC	\cGLS
\ABP	\ACP	\cGLSpl
\AS	\ACS	\GLSxtrshort
\ASP	\ACSP	\GLSxtrshortpl
\AL	\ACL	\GLSxtrlong
\ALP	\ACLP	\GLSxtrlongpl
\AF	\ACF	\GLSxtrfull
\AFP	\ACFP	\GLSxtrfullpl
\newabbr	\newabbr	\newabbreviation

is to simply do its argument, but in the glossary it's activated for those categories that have the `tagging` attribute set to "true". For those cases it will use:

```
\glsxtrtagfont{⟨text⟩}
```

This command defaults to `\underline{⟨text⟩}` but may be redefined as required.

The control sequence ⟨*cs*⟩ can't already be defined when used with the unstarred version of `\GlsXtrEnableInitialTagging` for safety reasons. The starred version will overwrite any previous definition of ⟨*cs*⟩. As with redefining any commands, ensure that you don't redefine something important.

The first argument of `\GlsXtrEnableInitialTagging` is a comma-separated list of category names. The `tagging` attribute will automatically be set to `true` for those categories. You can later set this attribute for other categories (see §10) but this must be done before the glossary is displayed.

Example 6 uses initial tagging for both the `acronym` and `abbreviation` categories. The custom command `\itag` is defined as the tagging command. 6

```
\makeglossaries
\GlsXtrEnableInitialTagging
 {acronym,abbreviation}{\itag}
\setabbreviationstyle[acronym]{short-nolong-desc}
\newacronym
 [description={a system for detecting the
 location and speed of ships, aircraft, etc, through
 the use of radio waves}% description of this term
 ]
 {radar}% identifying label
 {radar}% short form
 {\itag{ra}dio \itag{d}etection \itag{a}nd
  \itag{r}anging}

\newabbreviation{xml}{XML}
 {e\itag{x}tensible \itag{m}arkup \itag{l}anguage}

\newabbreviation[category={other}]{tne}{TNE}
 {\itag{t}agging \itag{n}ot \itag{e}nabled}

\begin{document}
First use: \gls{radar}, \gls{xml}, \gls{tne}.

Long form only: \glsxtrlong{radar},
```

```
\glsxtrlong{xml}, \glsxtrlong{tne}.
\printglossaries
\end{document}
```

Example 6: Tagging abbreviation initials

First use: radar, extensible markup language (XML), tagging not enabled (TNE).

Long form only: radio detection and ranging, extensible markup language, tagging not enabled.

Glossary

radar (radio detection and ranging) a system for detecting the location and speed of ships, aircraft, etc, through the use of radio waves 1

TNE tagging not enabled 1

XML extensible markup language 1

The underlining of the tagged letters only occurs in the glossary and then only for entries with the `tagging` attribute set.

4.5. Abbreviation Styles

The abbreviation style must be set before the abbreviation with the corresponding category is defined. If you are using `bib2gls`, the style must be set before `\GlsXtrLoadResources`.

The style for a particular category is set using:

`\setabbreviationstyle[`⟨*category*⟩`]{`⟨*style-name*⟩`}`

If the ⟨*category*⟩ is omitted, `abbreviation` is assumed. Remember that `\newacronym` sets the `category` to `acronym` so with `\newacronym` you need to change the style with:

```
\setabbreviationstyle[acronym]{⟨style-name⟩}
```

The style associated with the abbreviation category will be used if an abbreviation is defined with a category that doesn't have an associated style.

Once you have defined an abbreviation with a given category, you can't subsequently change the style for that category. You can't have more than one style per category. The default style for the abbreviation category is long-short and the default style for the acronym category is short-nolong.

Example 7 has a custom latin category which doesn't have an associated abbreviation style, so it uses the style assigned to the abbreviation category, not the acronym category. The only reason that the "radar" abbreviation (defined with \newacronym) uses the style associated with the acronym category is because the default definition of \newacronym sets category={acronym}. 7

```
\usepackage[T1]{fontenc}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-short-sc}
\newabbreviation{html}{html}
{hypertext markup language}
\setabbreviationstyle[acronym]{footnote}
\newacronym{radar}{radar}
{radio detection and ranging}
\newacronym[category={latin}]{ibid}{ibid}{ibidem}
\begin{document}
\gls{html}, \gls{radar} and \gls{ibid}.
\printunsrtglossaries
\end{document}
```

4.5.1. Predefined Abbreviation Styles

There are two types of abbreviation styles: those that treat the abbreviation as a regular entry (so that \gls uses \glsgenentryfmt and is encapsulated with \glsxtrregularfont) and those that don't treat the abbreviation as a regular entry (so that \gls uses \glsxtrgenabbrvfmt and is encapsulated with \glsxtrabbreviationfont). See §5.5.5 for further details of those commands.

The non-regular abbreviation styles allow for more complex formats than the regular styles.

Example 7: Category without an associated abbreviation style

hypertext markup language (HTML), radar[1] and ibidem (IBID).

Glossary

HTML hypertext markup language

radar radio detection and ranging

IBID ibidem

[1] radio detection and ranging

The regular entry abbreviation styles set the `regular` attribute to `true` for the category assigned to each abbreviation with that style. This means that on first use, `\gls` uses the value of the `first` field and on subsequent use `\gls` uses the value of the `text` field (and analogously for the plural and case-changing versions).

The non-regular abbreviation styles don't set the `regular` attribute, unless it has already been set, in which case it will be changed to `false`. The `first` and `text` fields (and their plural forms) are set, but they aren't used by commands like `\gls`, which instead use formatting commands, such as `\glsxtrfullformat` and `\glsxtrsubsequentfmt`, which are defined by the style.

In both cases, the first use of `\gls` may not match the text produced by `\glsfirst` (and likewise for the plural and case-changing versions).

The `short` and `long` fields are set as appropriate and may be accessed through commands like `\glsxtrshort` and `\glsxtrlong`. These may appear slightly differently to the way the short or long form is displayed within `\gls`, depending on the style.

The sample file `sample-abbr-styles.pdf` demonstrates all predefined styles described here.

For the "sc" styles that use `\textsc`, be careful about your choice of fonts as some only have limited support. For example, you may not be able to combine bold and small-caps. If you're using pdfLaTeX, I recommend that you at least use the fontenc package with the `T1` option or something similar.

The predefined styles have helper commands to make it easier to modify the format. These are described in §4.5.1.3.

Table 4.2 lists the nearest equivalent glossaries-extra abbreviation styles for the predefined acronym styles provided by glossaries, but note that the new styles use different formatting commands.

The example documents used to illustrate the predefined styles in the sub-sections below are all in the form (document class and options may vary):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\setabbreviationstyle{⟨style-name⟩}
\newabbreviation[⟨options⟩]{ex}{⟨short⟩}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
```

Table 4.2.: Base Acronym Styles `\setacronymstyle{`⟨*base-style-name*⟩`}` Verses New Abbreviation Styles `\setabbreviationstyle[`⟨*category*⟩`]{`⟨*new-style-name*⟩`}`

Base Style Name	**New Style Name**
long-sc-short	long-short-sc
long-sm-short	long-short-sm
long-sp-short	long-short with `\renewcommand{\glsxtrfullsep}{\glsabspace}`
short-long	short-long
sc-short-long	short-sc-long
sm-short-long	short-sm-long
long-short-desc	long-short-desc
long-sc-short-desc	long-short-sc-desc
long-sm-short-desc	long-short-sm-desc
long-sp-short-desc	long-short-desc with `\renewcommand{\glsxtrfullsep}{\glsabspace}`
short-long-desc	short-long-desc
sc-short-long-desc	short-sc-long-desc
sm-short-long-desc	short-sm-long-desc
dua	long-noshort
dua-desc	long-noshort-desc
footnote	short-footnote
footnote-sc	short-sc-footnote
footnote-sm	short-sm-footnote
footnote-desc	short-footnote-desc
footnote-sc-desc	short-sc-footnote-desc
footnote-sm-desc	short-sm-footnote-desc

```
\printunsrtglossaries
\end{document}
```

where ⟨*style-name*⟩ is the name of the abbreviation style, ⟨*short*⟩ is either "SHRT FM" or (for the small caps examples) "shrt fm". The styles that require the `description` or `user1` key to be set will include that in ⟨*options*⟩ otherwise the optional argument of `\newabbreviation` will be omitted. The examples with a style that requires `\textsmaller` will load relsize. The "hyphen" styles set the `markwords` and `markshortwords` attributes. Note that hyperref is loaded with the `colorlinks` option, so the hyperlink text will be red.

The naming scheme for abbreviation styles is as follows:

- ⟨*field1*⟩[−⟨*modifier1*⟩]−[post]⟨*field2*⟩[−⟨*modifier2*⟩][−user]

  This is for the parenthetical styles. The −⟨*modifier*⟩ parts may be omitted. These styles display ⟨*field1*⟩ followed by ⟨*field2*⟩ in parentheses. If ⟨*field1*⟩ or ⟨*field2*⟩ starts with "no" then that element is omitted from the display style (no parenthetical part) but is included in the inline style.

  If `post` is present then ⟨*field2*⟩ is placed after the link text using the post-link hook. Note that this will use the singular form of ⟨*field2*⟩ by default, even if `\glspl` is used. The corresponding non-post style will use the matching form for ⟨*field2*⟩.

  If the −⟨*modifier*⟩ part is present and is one of `sc`, `sm` or `em`, then the field has a font changing command applied to it.

  The modifier `−only` indicates that field is only present according to whether or not the entry has been used.

  The modifier `−hyphen` indicates the style will substitute inter-word spaces (that have been marked up with the `markwords` or `markshortwords` attributes) will be changed to spaces if the inserted material starts with a hyphen (but not for the set of `\glsxtrshort` and `\glsxtrlong` commands).

  If the `−user` part is present, then the value of the field given by `\glsxtruserfield{}{user1}`, if set, is inserted into the parenthetical material.

  Examples:

  - long−noshort−sc: ⟨*field1*⟩ is the long form, the short form is set in small caps but omitted in the display style.
  - long−em−short−em: both the long form and the short form are emphasized. The short form is in parentheses.
  - long−short−em: the short form is emphasized but not the long form. The short form is in parentheses.
  - long−short−user: if the `user1` key has been set, this produces the style ⟨*long*⟩ (⟨*short*⟩, ⟨*user1*⟩) otherwise it just produces ⟨*long*⟩ (⟨*short*⟩).
  - long−hyphen−postshort−hyphen: the short form and the inserted material (provided by the final optional argument of commands like `\gls`) is moved to the post-link

hook. The long form is formatted according to `\glslonghyphenfont` (or `\glsfirstlonghyphenfont` on first use). The short form is formatted according to `\glsabbrvhyphenfont` (or `\glsfirstabbrvhyphenfont` on first use).

- ⟨*style*⟩`-noreg`

  Some styles set the `regular` attribute. In some cases, there's a version of the style that doesn't set this attribute. For example, long-em-noshort-em sets the `regular` attribute. The long-em-noshort-em-noreg style is a minor variation of that style that sets the attribute to `false`.

  There are a few "noshort" styles, such as long-hyphen-noshort-noreg, where there isn't a corresponding regular version. This is because the style won't work properly with the `regular` attribute set, but the naming scheme is maintained for consistency with the other "noshort" styles.

- ⟨*field1*⟩[-⟨*modifier1*⟩]-[`post`]`footnote`

  The display style uses ⟨*field1*⟩ followed by a footnote with the other field in it. If `post` is present then the footnote is placed after the link text using the post-link hook. The inline style does ⟨*field1*⟩ followed by the other field in parentheses.

  If -⟨*modifier1*⟩ is present, ⟨*field1*⟩ has a font-changing command applied to it.

  Examples:

  - short-footnote: short form in the text with the long form in the footnote.
  - short-sc-postfootnote: short form in smallcaps with the long form in the footnote outside of the link text.

  Take care with the footnote styles. Remember that there are some situations where `\footnote` doesn't work.

- ⟨*style*⟩`-desc`

  Like ⟨*style*⟩ but the `description` key must be provided when defining abbreviations with this style.

  Examples:

  - short-long-desc: like short-long but requires a description.
  - short-em-footnote-desc: like short-em-footnote but requires a description.

Not all combinations that fit the above syntax are provided. Pre-version 1.04 styles that didn't fit this naming scheme are either provided with a synonym (where the former name wasn't ambiguous) or provided with a deprecated synonym (where the former name was confusing).

4.5.1.1. Regular Styles

The following abbreviation styles set the `regular` attribute to `true` for all categories that have abbreviations defined with any of these styles. This means that they are treated like ordinary entries and are encapsulated with `\glsxtrregularfont` not `\glsxtrabbreviationfont`. The `\gls`-like commands are formatted according to `\glsgenentryfmt`.

4.5.1.1.1. Short Styles

These styles only show the short form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.5 for style commands.

short-nolong

Only the short form is shown on first use of the `\gls`-like commands. The inline full form uses the same parenthetical style as short-long (`\glsxtrshortlongformat`). Font variations are available with short-sc-nolong, short-sm-nolong and short-em-nolong.

```
\setabbreviationstyle{short-nolong}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 8: The `short-nolong` abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

The long form is formatted with `\glslongdefaultfont` for the `\glsxtrlong` set of commands.

The short form is formatted with `\glsfirstabbrvdefaultfont` within the full form and with `\glsabbrvdefaultfont` for subsequent use and for the `\glsxtr-`

short set of commands.

The name is set to the short form (\glsxtrshortnolongname) and the description is set to the unencapsulated long form.

short *alias:* short−nolong

A synonym for short−nolong.

short−nolong−desc

As short−nolong but the description must be supplied in the optional argument of \newabbreviation. Font variations are available with short−sc−nolong−desc, short−sm−nolong−desc and short−em−nolong−desc.

```
\setabbreviationstyle{short-nolong-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 9: The short-nolong-desc abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM (long form) sample description

The name is set to the short form followed by the long form in parentheses (\glsxtrshortdescname), and the sort is set to just the short form.

short-desc *alias:* short-nolong-desc

A synonym for short-nolong-desc.

nolong-short

The same as short-nolong except for the inline full form, which shows the same parenthetical style as long-short (`\glsxtrlongshortformat`). Font variations are available with nolong-short-sc, nolong-short-sm and nolong-short-em.

```
\setabbreviationstyle{nolong-short}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 10: The `nolong-short` abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

short-sc-nolong

This style is like short-nolong but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sc-nolong}
```

```
\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 11: The `short-sc-nolong` abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

short–sc *alias:* short–sc–nolong

A synonym for short–sc–nolong.

short–sc–nolong–desc

This style is like short–nolong–desc but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sc-nolong-desc}

\newabbreviation[description={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 12: The short-sc-nolong-desc abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM (long form) sample description

short-sc-desc *alias:* short-sc-nolong-desc

A synonym for short-sc-nolong-desc.

nolong-short-sc

This style is like nolong-short but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

```
\setabbreviationstyle{nolong-short-sc}

\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 13: The nolong-short-sc abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

short-sm-nolong

This style is like short-nolong but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sm-nolong}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 14: The short-sm-nolong abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

short-sm *alias:* short-sm-nolong

A synonym for short-sm-nolong.

short-sm-nolong-desc

This style is like short-nolong-desc but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sm-nolong-desc}

\newabbreviation[description={sample description}]
```

```
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 15: The `short-sm-nolong-desc` abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM (long form) sample description

short−sm−desc *alias:* short−sm−nolong−desc

A synonym for short−sm−nolong−desc.

nolong−short−sm

This style is like nolong−short but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{nolong-short-sm}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 16: The `nolong-short-sm` abbreviation style

First: SHRT FM-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: SHRT FMs-insert. First no insert: SHRT FM.

Glossary

SHRT FM long form

short–em–nolong

This style is like short–nolong but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-em-nolong}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 17: The `short-em-nolong` abbreviation style

First: *SHRT FM*-insert. Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert. First no insert: *SHRT FM*.

Glossary

SHRT FM long form

short–em *alias:* short–em–nolong

A synonym for short–em–nolong.

short-em-nolong-desc

This style is like short-nolong-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-em-nolong-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 18: The `short-em-nolong-desc` abbreviation style

First: *SHRT FM*-insert. Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert. First no insert: *SHRT FM*.

Glossary

SHRT FM **(long form)** sample description

short-em-desc *alias:* short-em-nolong-desc

A synonym for short-em-nolong-desc.

nolong-short-em

This style is like nolong-short but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{nolong-short-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 19: The `nolong-short-em` abbreviation style

First: *SHRT FM*-insert. Next: *SHRT FM*-insert. Full: long form-insert (*SHRT FM*). First plural: *SHRT FMs*-insert. First no insert: *SHRT FM*.

Glossary

SHRT FM long form

4.5.1.1.2. Long Styles

These styles only show the long form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.6 for style commands.

long-noshort-desc

Only the long form is shown on first use and subsequent use of the `\gls`-like commands (`\glsxtrlongformat`). The inline full form uses the same parenthetical style as long-short (`\glsxtrlongshortformat`). Font variations are available with long-noshort-sc-desc, long-noshort-sm-desc and long-noshort-em-desc.

```
\setabbreviationstyle{long-noshort-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
```

```
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 20: The `long-noshort-desc` abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

The long form is formatted with `\glsfirstlongdefaultfont` on first use and `\glslongdefaultfont` for subsequent use and for the `\glsxtrlong` set of commands.

The short form is formatted with `\glsfirstabbrvdefaultfont` within the inline full form and with `\glsabbrvdefaultfont` for the `\glsxtrshort` set of commands.

The name is set to the long form (`\glsxtrlongnoshortdescname`) and the description must be supplied.

long-desc *alias:* long-noshort-desc

A synonym for long-noshort-desc.

long-noshort

This is like long-noshort-desc but the name is set to the short form (`\glsxtrlongnoshortname`) and the description is set to the unencapsulated long form.

```
\setabbreviationstyle{long-noshort}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
```

```
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 21: The `long-noshort` abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long *alias:* long−noshort

A synonym for long−noshort.

long−noshort−sc

This style is like long−noshort but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-sc}

\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 22: The `long-noshort-sc` abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long-noshort-sc-desc

This style is like long-noshort-desc but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-sc-desc}

\newabbreviation[description={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 23: The `long-noshort-sc-desc` abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

long-noshort-sm

This style is like long-noshort but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont`

and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-sm}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 24: The long-noshort-sm abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long-noshort-sm-desc

This style is like long-noshort-desc but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-sm-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
```

```
\end{document}
```

Example 25: The long-noshort-sm-desc abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

long-noshort-em

This style is like long-noshort but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 26: The long-noshort-em abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (*SHRT FM*). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long-noshort-em-desc

This style is like long-noshort-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-noshort-em-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 27: The `long-noshort-em-desc` abbreviation style

First: long form-insert. Next: long form-insert. Full: long form-insert (*SHRT FM*). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

long-em-noshort-em

This style is like long-noshort but it uses `\glsxtremsuffix`, `\glsabbrvemfont`, `\glsfirstabbrvemfont`, `\glslongemfont` and `\glsfirstlongemfont` (see §4.5.1.3.9). This emphasizes both the long and short forms.

```
\setabbreviationstyle{long-em-noshort-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
```

```
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 28: The `long-em-noshort-em` abbreviation style

First: *long form*-insert. Next: *long form*-insert. Full: *long form*-insert (*SHRT FM*). First plural: *long forms*-insert. First no insert: *long form*.

Glossary

SHRT FM *long form*

long-em-noshort-em-desc

This style is like long-noshort-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont`, `\glsfirstabbrvemfont`, `\glslongemfont` and `\glsfirstlongemfont` (see §4.5.1.3.9). This emphasizes both the long and short forms.

```
\setabbreviationstyle{long-em-noshort-em-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 29: The `long-em-noshort-em-desc` abbreviation style

First: *long form*-insert. Next: *long form*-insert. Full: *long form*-insert (*SHRT FM*). First plural: *long forms*-insert. First no insert: *long form*.

Glossary

long form sample description

4.5.1.2. Non-Regular Styles

The following abbreviation styles will set the `regular` attribute to `false` if it has previously been set. If it hasn't already been set, it's left unset. Other attributes may also be set, depending on the style.

The non-regular styles are too complicated to use `\glsgenentryfmt` as the display style (with the `\gls`-like commands). Instead they use `\glsxtrgenabbrvfmt`. This means that these styles won't work if you provide your own custom display style (using `\defglsentryfmt`) that doesn't check for the `regular` attribute.

Avoid using `\glsfirst`, `\glsfirstplural`, `\glstext` and `\glsplural` (or their case-changing variants) with these styles. There are also some styles that can be problematic with `\GLSname`.

4.5.1.2.1. Long (Short) Styles

These styles show the long form followed by the short form in parentheses on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.2 for style commands.

long-short

The ⟨*insert*⟩ is placed after the long form on first use of the `\gls`-like commands and after the short form on subsequent use. The inline full form is the same as the display full form (`\glsxtrlongshortformat`). Font variations are available with long-short-sc, long-short-sm, long-short-em and long-em-short-em.

```
\setabbreviationstyle{long-short}

\newabbreviation{ex}{SHRT FM}{long form}
```

```
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 30: The `long-short` abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

SHRT FM long form

The long form is formatted with `\glsfirstlongdefaultfont` within the full form and with `\glslongdefaultfont` for the `\glsxtrlong` set of commands.

The short form is formatted with `\glsfirstabbrvdefaultfont` within the full form and with `\glsabbrvdefaultfont` for subsequent use and for the `\glsxtrshort` set of commands.

The name is set to the short form (`\glsxtrlongshortname`) and the description is set to the unencapsulated long form.

long-short-desc

As long-short but the description must be supplied in the optional argument of `\newabbreviation`. Font variations are available with long-short-sc-desc, long-short-sm-desc, long-short-em-desc and long-em-short-em-desc.

```
\setabbreviationstyle{long-short-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
```

```
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 31: The `long-short-desc` abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

long form (SHRT FM) sample description

The name and sort are set to the long form followed by the short form in parentheses (`\glsxtrlongshortdescname` and `\glsxtrlongshortdescsort`).

long-short-sc

This style is like long-short but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-sc}

\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 32: The `long-short-sc` abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

SHRT FM long form

long-short-sc-desc

This style is like long-short-desc but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-sc-desc}

\newabbreviation[description={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 33: The `long-short-sc-desc` abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

long form (SHRT FM) sample description

long-short-sm

This style is like long-short but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-sm}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 34: The long-short-sm abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

SHRT FM long form

long-short-sm-desc

This style is like long-short-desc but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-sm-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
```

```
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 35: The `long-short-sm-desc` abbreviation style

First: long form-insert (SHRT FM). Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs). First no insert: long form (SHRT FM).

Glossary

long form (SHRT FM) sample description

long-short-em

This style is like long-short but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 36: The `long-short-em` abbreviation style

First: long form-insert (*SHRT FM*). Next: *SHRT FM*-insert. Full: long form-insert (*SHRT FM*). First plural: long forms-insert (*SHRT FMs*). First no insert: long form (*SHRT FM*).

Glossary

SHRT FM long form

long-short-em-desc

This style is like long-short-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{long-short-em-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 37: The `long-short-em-desc` abbreviation style

First: long form-insert (*SHRT FM*). Next: *SHRT FM*-insert. Full: long form-insert (*SHRT FM*). First plural: long forms-insert (*SHRT FMs*). First no insert: long form (*SHRT FM*).

Glossary

long form (*SHRT FM*) sample description

long-em-short-em

This style is like long-short but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont`, `\glsfirstlongemfont` and `\glslongemfont` (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

```
\setabbreviationstyle{long-em-short-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 38: The `long-em-short-em` abbreviation style

First: *long form*-insert (*SHRT FM*). Next: *SHRT FM*-insert. Full: *long form*-insert (*SHRT FM*). First plural: *long forms*-insert (*SHRT FMs*). First no insert: *long form* (*SHRT FM*).

Glossary

SHRT FM *long form*

long-em-short-em-desc

This style is like long-short-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont`, `\glsfirstlongemfont` and `\glslongemfont` (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

```
\setabbreviationstyle{long-em-short-em-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}
```

```
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 39: The `long-em-short-em-desc` abbreviation style

First: *long form*-insert (*SHRT FM*). Next: *SHRT FM*-insert. Full: *long form*-insert (*SHRT FM*). First plural: *long forms*-insert (*SHRT FMs*). First no insert: *long form* (*SHRT FM*).

Glossary

long form (***SHRT FM***) sample description

4.5.1.2.2. Long (Short, User) Styles

These styles are like the long (short) styles in §4.5.1.2.1 but additional content can be supplied in the field identified by `\glsxtruserfield`, which will be placed in the parenthetical content on first use (if set). The inline full form is the same as the display full form.

These styles use the commands `\glsxtrusersuffix`, `\glsabbrvuserfont`, `\glsfirstabbrvuserfont`, `\glslonguserfont` and `\glsfirstlonguserfont` (except where noted). See §4.5.1.3.1 and §4.5.1.3.3 for style commands.

If you need to change the font, you can redefine the associated commands (listed above). However, since small caps are awkward because the short plural suffix needs to counteract the small caps, small caps versions are provided.

long-short-user

This style is like long-short but it includes the additional content in the parentheses on first use or the inline full form. The description is obtained from `\glsuserdescription`, which can be redefined to include the additional information, if required.

```
\setabbreviationstyle{long-short-user}

\newabbreviation[user1={extra info}]{ex}{SHRT
FM}{long form}
```

```
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 40: The `long-short-user` abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert. Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FMs, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

SHRT FM long form

long-short-user-desc

This style is like long-short-user but the description must be supplied. The name is obtained from `\glsxtrlongshortuserdescname`.

```
\setabbreviationstyle{long-short-user-desc}

\newabbreviation[description={sample description},
 user1={extra info}]{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 41: The `long-short-user-desc` abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert. Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FMs, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

long form (SHRT FM, extra info) sample description

This style is incompatible with `\GLSname`.

If you need to use `\GLSname` with this style, you'll have to redefine `\glsxtrlongshortuserdescname` so that the field name doesn't include the entry label. For example:

```
\renewcommand{\glsxtrlongshortuserdescname}{%
 \protect\glslonguserfont{\the\glslongtok}%
 \space
 (\protect\glsabbrvuserfont{\the\glsshorttok})%
}
```

long-postshort-user

This style is like long-short-user but the parenthetical material is placed in the post-link hook. Note that, unlike long-short-user, the plural form isn't used in the parenthetical material. If you require this, you will need to redefine `\glsxtrpostusershortformat`.

```
\setabbreviationstyle{long-postshort-user}

\newabbreviation[user1={extra info}]{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
```

```
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 42: The `long-postshort-user` abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert.
Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert
(SHRT FM, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

SHRT FM long form

long-postshort-user-desc

This style is like long-postshort-user but the description must be supplied. The name is obtained from `\glsxtrlongshortuserdescname`.

```
\setabbreviationstyle{long-postshort-user-desc}

\newabbreviation[description={sample description},
 user1={extra info}]{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 43: The long-postshort-user-desc abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert.
Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert
(SHRT FM, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

long form (SHRT FM, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you'll have to redefine \glsxtrshortlonguserdescname so that the field name doesn't include the entry label, as for long-short-user-desc.

long-postshort-sc-user

This style is like long-postshort-user but it uses \glsxtrscusersuffix, \glsabbrvscuserfont and \glsfirstabbrvscuserfont. The name value is obtained from \glsxtrlongshortscusername.

```
\setabbreviationstyle{long-postshort-sc-user}

\newabbreviation[user1={extra info}]{ex}{shrt
fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 44: The `long-postshort-sc-user` abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert. Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FM, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

SHRT FM long form

long-postshort-sc-user-desc

This style is like long-postshort-sc-user but the description must be supplied. The name is obtained from `\glsxtrlongshortuserdescname`.

```
\setabbreviationstyle{long-postshort-sc-user-desc}

\newabbreviation[description={sample description},
 user1={extra info}]{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 45: The `long-postshort-sc-user-desc` abbreviation style

First: long form-insert (SHRT FM, extra info). Next: SHRT FM-insert. Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FM, extra info). First no insert: long form (SHRT FM, extra info).

Glossary

long form (SHRT FM, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you'll have to redefine \glsxtrlongshortscuserdescname so that the field name doesn't include the entry label.

4.5.1.2.3. Short (Long) Styles

These styles show the short form followed by the long form in parentheses on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.2 for style commands.

short-long

The ⟨*insert*⟩ is placed after the short form on first use and subsequent use of the \gls-like commands. The inline full form is the same as the display full form (\glsxtrshortlongformat). Font variations are available with short-sc-long, short-sm-long, short-em-long and short-em-long-em.

```
\setabbreviationstyle{short-long}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 46: The short-long abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM long form

The long form is formatted with \glsfirstlongdefaultfont within the full form and with \glslongdefaultfont for the \glsxtrlong set of commands.

The short form is formatted with \glsfirstabbrvdefaultfont within the full

form and with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.

The name is set to the short form (\glsxtrlongshortname) and the description is set to the unencapsulated long form.

short-long-desc

As short-long but the description must be supplied in the optional argument of \newabbreviation. Font variations are available with short-sc-long-desc, short-sm-long-desc, short-em-long-desc and short-em-long-em-desc.

```
\setabbreviationstyle{short-long-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 47: The short-long-desc abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM (long form) sample description

The name is set to the short form followed by the long form in parentheses (\glsxtrshortlongdescname), and the sort is set to just the short form (\glsxtrshortlongdescsort).

short-sc-long

This style is like short-long but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sc-long}

\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 48: The `short-sc-long` abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM long form

short-sc-long-desc

This style is like short-long-desc but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sc-long-desc}

\newabbreviation[description={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 49: The short-sc-long-desc abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM (long form) sample description

short-sm-long

This style is like short-long but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sm-long}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 50: The short-sm-long abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM long form

short-sm-long-desc

This style is like short-long-desc but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sm-long-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 51: The `short-sm-long-desc` abbreviation style

First: SHRT FM-insert (long form). Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert (long forms). First no insert: SHRT FM (long form).

Glossary

SHRT FM (long form) sample description

short-em-long

This style is like short-long but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-em-long}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
```

```
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 52: The `short-em-long` abbreviation style

First: *SHRT FM*-insert (long form). Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert (long forms). First no insert: *SHRT FM* (long form).

Glossary

SHRT FM long form

short-em-long-desc

This style is like short-long-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle{short-em-long-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 53: The short-em-long-desc abbreviation style

First: *SHRT FM*-insert (long form). Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert (long forms). First no insert: *SHRT FM* (long form).

Glossary

SHRT FM **(long form)** sample description

short-em-long-em

This style is like short-long but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont, \glsfirstlongemfont and \glslongemfont (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

```
\setabbreviationstyle{short-em-long-em}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 54: The short-em-long-em abbreviation style

First: *SHRT FM*-insert (*long form*). Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (*long form*). First plural: *SHRT FMs*-insert (*long forms*). First no insert: *SHRT FM* (*long form*).

Glossary

SHRT FM *long form*

short-em-long-em-desc

This style is like short-long-desc but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont, \glsfirstlongemfont and \glslongemfont (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

```
\setabbreviationstyle{short-em-long-em-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 55: The short-em-long-em-desc abbreviation style

First: *SHRT FM*-insert (*long form*). Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (*long form*). First plural: *SHRT FMs*-insert (*long forms*). First no insert: *SHRT FM* (*long form*).

Glossary

SHRT FM (***long form***) sample description

4.5.1.2.4. Short (Long, User) Styles

These styles are like the short (long) styles in §4.5.1.2.3 but additional content can be supplied in the field identified by \glsxtruserfield, which will be placed in the parenthetical content on first use (if set). The inline full form is the same as the display full form.

These styles use the commands \glsxtrusersuffix, \glsabbrvuserfont, \glsfirstabbrvuserfont, \glslonguserfont and \glsfirstlonguserfont (except where noted). See §4.5.1.3.1 and §4.5.1.3.3 for style commands.

short-long-user

This style is like short-long but it includes the additional content in the parentheses on first use

or the inline full form.

The description is obtained from \glsuserdescription, which can be redefined to include the additional information, if required.

```
\setabbreviationstyle{short-long-user}

\newabbreviation[user1={extra info}]{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 56: The short-long-user abbreviation style

First: SHRT FM-insert (long form, extra info). Next: SHRT FM-insert. Full: SHRT FM-insert (long form, extra info). First plural: SHRT FMs-insert (long forms, extra info). First no insert: SHRT FM (long form, extra info).

Glossary

SHRT FM long form

short-long-user-desc

This style is like short-long-user but the description must be provided. The name is obtained from \glsxtrshortlonguserdescname and the sort value is obtained from \glsxtrshortlongdescsort.

```
\setabbreviationstyle{short-long-user-desc}

\newabbreviation[description={sample description},
 user1={extra info}]{ex}{SHRT FM}{long form}
```

```
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 57: The `short-long-user-desc` abbreviation style

First: SHRT FM-insert (long form, extra info). Next: SHRT FM-insert. Full: SHRT FM-insert (long form, extra info). First plural: SHRT FMs-insert (long forms, extra info). First no insert: SHRT FM (long form, extra info).

Glossary

SHRT FM (long form, extra info) sample description

This style is incompatible with `\GLSname`.

If you need to use `\GLSname` with this style, you'll have to redefine `\glsxtrshortlonguserdescname` so that the field name doesn't include the entry label. For example:

```
\renewcommand{\glsxtrlongshortuserdescname}{%
 \protect\glsabbrvuserfont{\the\glsshorttok}%
 \space
 (\protect\glslonguserfont{\the\glslongtok})%
}
```

short-postlong-user

This style is like short-long but it includes the additional content in the parentheses on first use or the inline full form. The parenthetical content is placed in the post-link hook which can avoid overly long hyperlinks.

The description is obtained from `\glsuserdescription`, which can be redefined to include the additional information, if required.

```
\setabbreviationstyle{short-postlong-user}

\newabbreviation[user1={extra info}]{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 58: The `short-postlong-user` abbreviation style

First: SHRT FM-insert (long form, extra info). Next: SHRT FM-insert. Full: SHRT FM-insert (long form, extra info). First plural: SHRT FMs-insert (long form, extra info). First no insert: SHRT FM (long form, extra info).

Glossary

SHRT FM long form

short-postlong-user-desc

This style is like short-postlong-user but the description must be provided. The name is obtained from `\glsxtrshortlonguserdescname`. The sort value is the short form.

```
\setabbreviationstyle{short-postlong-user-desc}

\newabbreviation[description={sample description},
 user1={extra info}]{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
```

```
\printunsrtglossaries
\end{document}
```

Example 59: The short-postlong-user-desc abbreviation style

First: SHRT FM-insert (long form, extra info). Next: SHRT FM-insert. Full: SHRT FM-insert (long form, extra info). First plural: SHRT FMs-insert (long form, extra info). First no insert: SHRT FM (long form, extra info).

Glossary

SHRT FM (long form, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you'll have to redefine \glsxtrshortlonguserdescname so that the field name doesn't include the entry label, as for short-long-user-desc.

4.5.1.2.5. Hyphen Styles

These styles test if the inserted material start with a hyphen. See §4.5.1.3.1, §4.5.1.3.2 and §4.5.1.3.7 for style commands.

These styles are designed to be used with the markwords attribute and (if the short form has spaces) the markshortwords attribute. If the inserted material starts with a hyphen, the spaces will be replaced with hyphens. This replacement won't take place if the corresponding attribute wasn't used to mark the inter-word spaces.

Note that \glsxtrshort and \glsxtrlong (and their plural and case-changing variants) don't perform the inter-word space substitution. The inline full form is slightly different from the display full form for the "post" styles.

long-hyphen-short-hyphen

This style is like long-short but checks the inserted material for a leading hyphen. The description is the long form encapsulated with \glslonghyphenfont. The name is obtained from \glsxtrlongshortname, and the sort value is obtained from \glsxtrlonghyphenshortsort. The inline full form is the same as the display full form.

```
\setabbreviationstyle{long-hyphen-short-hyphen}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 60: The `long-hyphen-short-hyphen` abbreviation style

First: long-form-insert (SHRT-FM-insert). Next: SHRT-FM-insert. Full: long-form-insert (SHRT-FM-insert). First plural: long-forms-insert (SHRT-FMs-insert). First no insert: long form (SHRT FM).

Glossary

SHRT FM long form

long-hyphen-postshort-hyphen

This style is like long-hyphen-short-hyphen but places the insert and parenthetical material in the post-link hook for the display full form.

```
\setabbreviationstyle{long-hyphen-postshort-hyphen}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
```

```
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 61: The long-hyphen-postshort-hyphen abbreviation style

First: long-form-insert (SHRT-FM-insert). Next: SHRT-FM-insert. Full: long form-insert (SHRT FM). First plural: long-forms-insert (SHRT-FM-insert). First no insert: long form (SHRT FM).

Glossary

SHRT FM long form

Note that the inline full form (\glsxtrfull) doesn't show the insert in the post-link hook, but instead places it at the end of the link text. This is because only the \gls-like commands (not the \glstext-like commands) set the placeholder \glsinsert to the supplied insert. If you want the insert to show in the parenthetical part of the post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert:

```
\renewcommand*{\glsxtrfullsaveinsert}[2]{%
 \def\glsinsert{#2}}
```

long-hyphen-short-hyphen-desc

This style is like long-hyphen-short-hyphen but the description must be supplied. The name is obtained from \glsxtrlongshortdescname, and the sort value is obtained from \glsxtrlongshortdescsort.

```
\setabbreviationstyle{long-hyphen-short-hyphen-desc}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
```

```
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 62: The `long-hyphen-short-hyphen-desc` abbreviation style

First: long-form-insert (SHRT-FM-insert). Next: SHRT-FM-insert. Full: long-form-insert (SHRT-FM-insert). First plural: long-forms-insert (SHRT-FMs-insert). First no insert: long form (SHRT FM).

Glossary

long form (SHRT FM) sample description

long-hyphen-postshort-hyphen-desc

This style is like long-hyphen-short-hyphen-desc but places the insert and parenthetical material in the post-link hook for the display full form.

```
\setabbreviationstyle
{long-hyphen-postshort-hyphen-desc}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 63: The long-hyphen-postshort-hyphen-desc abbreviation style

First: long-form-insert (SHRT-FM-insert). Next: SHRT-FM-insert. Full: long form-insert (SHRT FM). First plural: long-forms-insert (SHRT-FM-insert). First no insert: long form (SHRT FM).

Glossary

long form (SHRT FM) sample description

Note that as with the long-hyphen-postshort-hyphen style, the insert isn't included in the post-link hook by default for the inline full form. If you want the insert to show in the post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert.

long-hyphen-noshort-desc-noreg

This style is like long-noshort-desc-noreg but checks the inserted material for a leading hyphen. The description must be supplied. The name is obtained from \glsxtrlongnoshortdescname, and the sort value is obtained from \glsxtrlonghyphennoshortdescsort.

```
\setabbreviationstyle
{long-hyphen-noshort-desc-noreg}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 64: The long-hyphen-noshort-desc-noreg abbreviation style

First: long-form-insert. Next: long-form-insert. Full: long-form-insert (SHRT-FM-insert). First plural: long-forms-insert. First no insert: long form.

Glossary

long form sample description

long-hyphen-noshort-noreg

This style is like long-noshort-noreg but checks the inserted material for a leading hyphen. The description is set to the unencapsulated long form. The name is obtained from \glsxtrlongnoshortname, and the sort value is obtained from \glsxtrlonghyphennoshortsort.

```
\setabbreviationstyle{long-hyphen-noshort-noreg}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 65: The long-hyphen-noshort-noreg abbreviation style

First: long-form-insert. Next: long-form-insert. Full: long-form-insert (SHRT-FM-insert). First plural: long-forms-insert. First no insert: long form.

Glossary

SHRT FM long form

short–hyphen–long–hyphen

This style is like short–long but checks the inserted material for a leading hyphen. The description is the long form encapsulated with \glslonghyphenfont. The name is obtained from \glsxtrshortlongname and the sort value is obtained from \glsxtrshorthyphenlongsort.

```
\setabbreviationstyle{short-hyphen-long-hyphen}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 66: The short-hyphen-long-hyphen abbreviation style

First: SHRT-FM-insert (long-form-insert). Next: SHRT-FM-insert. Full: SHRT-FM-insert (long-form-insert). First plural: SHRT-FMs-insert (long-forms-insert). First no insert: SHRT FM (long form).

Glossary

SHRT FM long form

short–hyphen–postlong–hyphen

This style is like short–hyphen–long–hyphen but the insert and parenthetical material are placed in the post-link hook for the display full form.

```
\setabbreviationstyle{short-hyphen-postlong-hyphen}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 67: The short-hyphen-postlong-hyphen abbreviation style

First: SHRT-FM-insert (long-form-insert). Next: SHRT-FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT-FMs-insert (long-form-insert). First no insert: SHRT FM (long form).

Glossary

SHRT FM long form

Note that as with the long–hyphen–postshort–hyphen style, the insert isn't included in the

post-link hook by default for the inline full form. If you want the insert to show in the post-link hook for the inline full form you need to redefine `\glsxtrfullsaveinsert` (as described above, for the long-hyphen-postshort-hyphen style).

short-hyphen-long-hyphen-desc

This style is like short-hyphen-long-hyphen but the description must be supplied. The name is obtained from `\glsxtrshortlongdescname`, and the sort is obtained from `\glsxtrshortlongdescsort`.

```
\setabbreviationstyle{short-hyphen-long-hyphen-desc}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 68: The `short-hyphen-long-hyphen-desc` abbreviation style

First: SHRT-FM-insert (long-form-insert). Next: SHRT-FM-insert. Full: SHRT-FM-insert (long-form-insert). First plural: SHRT-FMs-insert (long-forms-insert). First no insert: SHRT FM (long form).

Glossary

SHRT FM (long form) sample description

short-hyphen-postlong-hyphen-desc

This style is like short-hyphen-long-hyphen-desc but the insert and parenthetical material are placed in the post-link hook for the display full form.

```
\setabbreviationstyle
{short-hyphen-postlong-hyphen-desc}

\glssetcategoryattributes{abbreviation}
 {markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 69: The `short-hyphen-postlong-hyphen-desc` abbreviation style

First: SHRT-FM-insert (long-form-insert). Next: SHRT-FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT-FMs-insert (long-form-insert). First no insert: SHRT FM (long form).

Glossary

SHRT FM (long form) sample description

Note that as with the long-hyphen-postshort-hyphen style, the insert isn't included in the post-link hook by default for the inline full form. If you want the insert to show in the post-link hook for the inline full form you need to redefine `\glsxtrfullsaveinsert` (as described above, for the long-hyphen-postshort-hyphen style).

4.5.1.2.6. Only Styles

These styles only show the long form on first use and only show the short form on subsequent use. The inline full form is the same as the display full form. See §4.5.1.3.1, §4.5.1.3.2 and §4.5.1.3.8 for style commands.

The inline full form uses a parenthetical style with the long form followed by the short form in parentheses.

long-only-short-only

The name is obtained from `\glsxtronlyname` and the sort value is just the short form.

The description is the long form encapsulated with \glslongonlyfont.

```
\setabbreviationstyle{long-only-short-only}

\newabbreviation{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 70: The long-only-short-only abbreviation style

First: long form-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long-only-short-only-desc

This is like long-only-short-only but the description must be supplied. The name is obtained from \glsxtronlydescname and the sort is obtained from \glsxtronlydescsort.

```
\setabbreviationstyle{long-only-short-only-desc}

\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
```

```
\printunsrtglossaries
\end{document}
```

Example 71: The long-only-short-only-desc abbreviation style

First: long form-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

long-only-short-sc-only

This is like long-only-short-only but uses small caps. The name is obtained from \glsxtrsconlyname, and it uses \glsabbrvsconlyfont, \glsfirstabbrvsconlyfont and \glsxtrsconlysuffix for the abbreviation fonts and plural suffix.

```
\setabbreviationstyle{long-only-short-sc-only}

\newabbreviation{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 72: The long-only-short-sc-only abbreviation style

First: long form-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

SHRT FM long form

long-only-short-sc-only-desc

This is like long-only-short-only-desc but uses small caps. The name is obtained from \glsxtrsconlydescname, and the sort is obtained from \glsxtrsconlydescsort.

```
\setabbreviationstyle{long-only-short-sc-only-desc}

\newabbreviation[description={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
First plural: \glspl[prereset]{ex}[-insert].
First no insert: \gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 73: The long-only-short-sc-only-desc abbreviation style

First: long form-insert. Next: SHRT FM-insert. Full: long form-insert (SHRT FM). First plural: long forms-insert. First no insert: long form.

Glossary

long form sample description

4.5.1.2.7. Footnote Styles

These styles show the short form (\glsxtrshortformat) with the long form as a footnote on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.4 for style commands.

The inline full form uses the same parenthetical style as short-long (\glsxtrshortlongformat). Font variations are available with short-sc-footnote, short-sm-footnote and short-em-footnote.

short-footnote

The ⟨*insert*⟩ is placed after the short form on first use and subsequent use of the \gls-like commands.

```
\setabbreviationstyle
{short-footnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 74: The short-footnote abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM long form

[1] long form
[2] long forms
[3] long form

The long form is formatted with \glsfirstlongfootnotefont within the full form and with \glslongfootnotefont for the \glsxtrlong set of commands.

The short form is formatted with \glsfirstabbrvdefaultfont within the full form and with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.

The name is set to the short form (\glsxtrfootnotename) and the description is set to the unencapsulated long form.

This style automatically sets the nohyperfirst attribute to true for the entry's category.

footnote *alias:* short-footnote

A synonym for short-footnote.

short-footnote-desc

As short-footnote but the description must be supplied in the optional argument of \newabbreviation.

```
\setabbreviationstyle
{short-footnote-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 75: The short-footnote-desc abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long forms
[3] long form

The name is set to the short form followed by the long form in parentheses (\glsxtrfootnotedescname), and the sort is set to just the short form (\glsxtrfootnotedescsort).

footnote-desc *alias:* short-footnote-desc

A synonym for short-footnote-desc.

short-postfootnote

Similar to short-footnote but the footnote command is placed in the post-link hook. This avoids the problem of nested hyperlinks caused by the footnote marker hyperlink being inside the link text (which is why the short-footnote style switches on the nohyperfirst attribute). Using the post-link hook makes it possible to check for following punctuation so that the footnote marker can be shifted after the punctuation character.

```
\setabbreviationstyle
{short-postfootnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 76: The `short-postfootnote` abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM long form

[1] long form
[2] long form
[3] long form

postfootnote *alias:* short−postfootnote

A synonym for short−postfootnote.

short−postfootnote−desc

Similar to short−footnote−desc but the footnote command is placed in the post-link hook as with short−postfootnote.

```
\setabbreviationstyle
{short-postfootnote-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 77: The `short-postfootnote-desc` abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long form
[3] long form

postfootnote−desc *alias:* short−postfootnote−desc

A synonym for short−postfootnote−desc.

short−sc−footnote

This style is like short−footnote but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sc-footnote}

\newabbreviation{ex}{shrt
fm}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 78: The `short-sc-footnote` abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM long form

[1] long form
[2] long forms
[3] long form

short-sc-footnote-desc

This style is like short-footnote-desc but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sc-footnote-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 79: The `short-sc-footnote-desc` abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long forms
[3] long form

short-sc-postfootnote

This style is like short-postfootnote but it uses `\glsxtrscsuffix`, `\glsabbrvscfont` and `\glsfirstabbrvscfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sc-postfootnote}

\newabbreviation{ex}{shrt
fm}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 80: The short-sc-postfootnote abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM long form

[1] long form
[2] long form
[3] long form

short-sc-postfootnote-desc

This style is like short-postfootnote-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sc-postfootnote
-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{shrt fm}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 81: The `short-sc-postfootnote-desc` abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long form
[3] long form

short-sm-footnote

This style is like short-footnote but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sm-footnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 82: The `short-sm-footnote` abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM long form

[1] long form
[2] long forms
[3] long form

short-sm-footnote-desc

This style is like short-footnote-desc but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sm-footnote-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 83: The `short-sm-footnote-desc` abbreviation style

First: SHRT FM-insert[1]. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert[2]. First no insert: SHRT FM[3].

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long forms
[3] long form

short-sm-postfootnote

This style is like short-postfootnote but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sm-postfootnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 84: The `short-sm-postfootnote` abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM long form

[1] long form
[2] long form
[3] long form

short-sm-postfootnote-desc

This style is like short-postfootnote-desc but it uses `\glsxtrsmsuffix`, `\glsabbrvsmfont` and `\glsfirstabbrvsmfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-sm-postfootnote
-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 85: The `short-sm-postfootnote-desc` abbreviation style

First: SHRT FM-insert.[1] Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert.[2] First no insert: SHRT FM.[3]

Glossary

SHRT FM (long form) sample description

[1] long form
[2] long form
[3] long form

short-em-footnote

This style is like short-footnote but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-em-footnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 86: The `short-em-footnote` abbreviation style

First: *SHRT FM*-insert[1]. Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert[2]. First no insert: *SHRT FM*[3].

Glossary

SHRT FM long form

[1] long form
[2] long forms
[3] long form

short-em-footnote-desc

This style is like short-footnote-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-em-footnote-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 87: The `short-em-footnote-desc` abbreviation style

First: *SHRT FM*-insert[1]. Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert[2]. First no insert: *SHRT FM*[3].

Glossary

SHRT FM **(long form)** sample description

[1] long form
[2] long forms
[3] long form

short-em-postfootnote

This style is like short-postfootnote but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-em-postfootnote}

\newabbreviation{ex}{SHRT
FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 88: The `short-em-postfootnote` abbreviation style

First: *SHRT FM-insert*.[1] Next: *SHRT FM-insert*. Full: *SHRT FM-insert (long form)*. First plural: *SHRT FMs-insert*.[2] First no insert: *SHRT FM*.[3]

Glossary

SHRT FM long form

[1] long form
[2] long form
[3] long form

short-em-postfootnote-desc

This style is like short-postfootnote-desc but it uses `\glsxtremsuffix`, `\glsabbrvemfont` and `\glsfirstabbrvemfont` (see §4.5.1.3.9).

```
\setabbreviationstyle
{short-em-postfootnote
-desc}

\newabbreviation[descrip-
tion={sample description}]
{ex}{SHRT FM}{long form}

\begin{document}
First: \gls{ex}[-insert].
Next: \gls{ex}[-insert].
Full:
\glsxtrfull{ex}[-insert].
First plural: \glspl
[prereset]{ex}[-insert].
First no insert:
\gls[prereset]{ex}.
\printunsrtglossaries
\end{document}
```

Example 89: The `short-em-postfootnote-desc` abbreviation style

First: *SHRT FM*-insert.[1] Next: *SHRT FM*-insert. Full: *SHRT FM*-insert (long form). First plural: *SHRT FMs*-insert.[2] First no insert: *SHRT FM*.[3]

Glossary

SHRT FM **(long form)** sample description

[1] long form
[2] long form
[3] long form

4.5.1.2.8. Short Styles

These styles only show the short form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.5 for style commands. They are essentially identical to the corresponding regular style listed in §4.5.1.2.8 except that they change the `regular` attribute to `false`.

short-nolong-noreg

This style is a non-regular version of the short-nolong style.

short-nolong-desc-noreg

This style is a non-regular version of the short-nolong-desc style.

nolong-short-noreg

This style is a non-regular version of the nolong-short style.

4.5.1.2.9. Long Styles

These styles only show the long form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.6 for style commands. They are essentially identical to the corresponding regular style listed in §4.5.1.2.9 except that they change the `regular` attribute to `false`.

long-noshort-desc-noreg

This style is a non-regular version of the long-noshort-desc style.

long-noshort-noreg

This style is a non-regular version of the long-noshort style.

long-em-noshort-em-noreg

This style is a non-regular version of the long-em-noshort-em style.

long-em-noshort-em-desc-noreg

This style is a non-regular version of the long-em-noshort-em-desc style.

4.5.1.3. Formatting Commands and Hooks

These commands are used by the predefined abbreviation styles. These are considered user commands, which you can redefine to customize the style.

4.5.1.3.1. General

These commands apply to all styles.

`\ifglsxtrinsertinside` ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`

This conditional is used to determine whether the ⟨*insert*⟩ part should go inside or outside of the style's font formatting commands. The default setting is false.

`\glsxtrinsertinsidetrue`

Set the insert inside conditional to true.

`\glsxtrinsertinsidefalse`

Set the insert inside conditional to false.

`\glsxtrparen{`⟨*text*⟩`}`

Used for parenthetical content in the inline full form and also, for some styles, the display full form. Note that this formats the opening and closing parentheses according to the inner formatting, but not the argument, which should already incorporate it. The default definition is:

```
\newcommand*{\glsxtrparen}[1]{%
 \glsxtrgenentrytextfmt{(}#1%
 \glsxtrgenentrytextfmt{)}}
```

\glsxtrfullsep{⟨*entry-label*⟩}

Separator placed before \glsxtrparen. This is a space by default, but it includes the inner formatting. The argument (the entry label) is ignored by default:

```
\newcommand*{\glsxtrfullsep}[1]{%
 \glsxtrgenentrytextfmt{ }}
```

You can redefine this to use \glsabspace if you want to have a non-breakable space if the short form is less than \glsacspacemax in width. (You can use \glsacspace instead, but note that \glsacspace doesn't incorporate the inner formatting.)

\glsabbrvdefaultfont{⟨*text*⟩}

Abbreviation font command used by styles that don't have specific font markup (for example, long-short but not long-em-short-em). This just does its argument.

\glsfirstabbrvdefaultfont{⟨*text*⟩}

First use abbreviation font command used by styles that don't have specific font markup. This is defined to just use \glsabbrvdefaultfont.

\glsxtrdefaultrevert{⟨*text*⟩}

This is the default definition of \glsxtrrevert used by styles that don't have specific font markup. If you redefine \glsabbrvdefaultfont, you will need to redefine \glsxtrdefaultrevert as applicable.

\glslongdefaultfont{⟨*text*⟩}

Long form font command used by styles that don't have specific font markup. This just does its argument.

```
\glsfirstlongdefaultfont{⟨text⟩}
```

First use long form font command used by styles that don't have specific font markup. This is defined to just use `\glslongdefaultfont`.

4.5.1.3.2. Parenthetical Styles

These commands apply to the parenthetical styles, such as long-short.

```
\glsxtrlongshortname
```

This command should expand to the value of the `name` key for styles like long-short. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
```

```
\glsxtrlongshortdescsort
```

This command should expand to the sort value used by styles such as long-short-desc. The default definition is:

```
\expandonce\glsxtrorglong\space
 (\expandonce\glsxtrorgshort)
```

Note that this uses the original ⟨*long*⟩ and ⟨*short*⟩ values supplied to `\newabbreviation`.

This command is irrelevant with the "unsrt" family of commands.

```
\glsxtrlongshortdescname
```

This command should expand to the name value used by styles such as long-short-desc. The default definition is:

```
\glsxplongfont{\the\glslongtok}{\glscategorylabel}%
\protect\glsxtrfullsep{\the\glslabeltok}%
\protect\glsxtrparen
```

```
  {\glsxpabbrvfont{\the\glsshorttok}
    {\glscategorylabel}}
```

This essentially expands to ⟨*long*⟩ (⟨*short*⟩) but includes the style font changing commands, the inner text formatting, and accessibility support.

```
\glsxtrshortlongname
```

This command should expand to the value of the `name` key for styles like short–long. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
```

```
\glsxtrshortlongdescsort
```

This command should expand to the value of the `sort` key for styles like short–long–desc. The default definition is just `\expandonce\glsxtrorgshort`.

This command is irrelevant with the "unsrt" family of commands.

```
\glsxtrshortlongdescname
```

This command should expand to the value of the `name` key for styles like short–long–desc. The default definition is:

```
\glsxpabbrvfont
 {\the\glsshorttok}{\glscategorylabel}%
\protect\glsxtrfullsep{\the\glslabeltok}%
\protect\glsxtrparen
  {\glsxplongfont{\the\glslongtok}
    {\glscategorylabel}}
```

4.5.1.3.3. User Styles

These commands apply to the "user" styles, such as long–short–user.

\glsxtruserfield

This command should expand to the internal label of the field used to store the additional information that should be shown in the parenthetical material on first use. The default is useri, which corresponds to the user1 key.

\glsxtruserparensep

The separator used within the parenthetical content. This defaults to a comma followed by a space.

\glsxtruserfieldfmt{⟨*text*⟩}

Used to encapsulate the value of the field given by \glsxtruserfield within \glsxtruserparen and \GLSxtruserparen. This simply does its argument by default.

The inner formatting with both \glsxtruserparen and \GLSxtruserparen, and the case-change with the latter, will be included in the argument of \glsxtruserfieldfmt.

For example, to emphasize the user value and separate it with a semi-colon instead of a comma:

```
\renewcommand{\glsxtruserparensep}{; }
\renewcommand{\glsxtruserfieldfmt}[1]{\emph{#1}}
```

\glsabbrvuserfont{⟨*text*⟩}

Formatting for the "user" short form. This defaults to \glsabbrvdefaultfont.

\glsfirstabbrvuserfont{⟨*text*⟩}

Formatting for the "user" short form shown on first use. This defaults to \glsabbrvuserfont.

\glsxtrusersuffix *initial:* \glsxtrabbrvpluralsuffix

Short plural suffix used by the "user" styles. This defaults to \glsxtrabbrvpluralsuffix.

\glslonguserfont{⟨*text*⟩}

Formatting for the "user" long form. This defaults to \glsabbrvdefaultfont.

\glsfirstlonguserfont{⟨text⟩}

Formatting for the "user" short form shown on first use. This defaults to \glslonguserfont.

\glsabbrvscuserfont{⟨text⟩}

Formatting for the "sc-user" short form. This uses \glsabbrvscfont, which in turn uses \textsc to apply a small caps style, so your document font needs to support it.

\textsc uses small capital glyphs for lowercase characters. Uppercase characters show as normal capitals. This means that you need to use lowercase characters in the abbreviation.

\glsfirstabbrvscuserfont{⟨text⟩}

Formatting for the "sc-user" short form shown on first use. This defaults to \glsabbrvscuserfont.

\glsxtrscuserrevert{⟨text⟩}

Counteracts the effect of \glsabbrvscuserfont. The default is \glsxtrscrevert. If you redefine \glsabbrvscuserfont, you will need to redefine \glsxtrscuserrevert as applicable.

\glsxtrscusersuffix

Short plural suffix used by the "sc-user" styles. This defaults to \glsxtrscsuffix.

\glsuserdescription{⟨text⟩}{⟨entry-label⟩}

The description field is set to this, where the ⟨text⟩ argument is the long form, for the "user" styles where the description is set by default. This is defined to ignore its second argument:

```
\newcommand*{\glsuserdescription}[2]{%
 \glslonguserfont{#1}}
```

If you want to include the information contained in the field identified by \glsxtruserfield, the second argument provides a way of accessing that field without relying on the \glscurrententrylabel placeholder. For example:

```
\renewcommand*{\glsuserdescription}[2]{%
 \glslonguserfont{#1}%
 \ifglshasfield{\glsxtruserfield}{#2}%
 {, \glscurrentfieldvalue}%
 {}%
}
```

\glsxtruserparen{⟨*text*⟩}{⟨*entry-label*⟩}

If the field given by \glsxtruserfield has been set, this essentially does:

\glsxtrfullsep{⟨*entry-label*⟩}\glsxtrparen{⟨*text*⟩, ⟨*user-value*⟩}

otherwise it does:

\glsxtrfullsep{⟨*entry-label*⟩}\glsxtrparen{⟨*text*⟩}

It's a little more complicated than this as the definition includes the inner formatting around the comma and the field value (⟨*user-value*⟩). The comma separator is given by \glsxtruserparensep, and the field value is encapsulated with \glsxtruserfieldfmt (with the inner formatting inside).

If you redefine this command, you will also need to redefine the following one in a similar manner.

\GLSxtruserparen{⟨*text*⟩}{⟨*entry-label*⟩}

As above but the content of the field given by \glsxtruserfield is converted to all caps. Note that simply applying an uppercase command to \glsxtruserparen can fail as it can cause the label to be converted to all caps, which is the reason why a separate command to internally perform the case-change is provided.

\glsxtrlongshortuserdescname

Expands to the value for the name key for styles like long-short-user-desc. The default definition is:

```
\protect\glslonguserfont{\the\glslongtok}%
\protect\glsxtruserparen
{\protect\glsabbrvuserfont{\the\glsshorttok}}
{\the\glslabeltok}
```

```
\glsxtrlongshortscusername
```

Expands to the value for the name key for styles like long-postshort-sc-user styles where the description is automatically set. The default definition is:

```
\protect\glsabbrvscuserfont{\the\glsshorttok}
```

```
\glsxtrlongshortscuserdescname
```

Expands to the value for the name key for styles like long-postshort-sc-user-desc. The default definition is:

```
\protect\glslonguserfont{\the\glslongtok}%
\protect\glsxtruserparen
{\protect\glsabbrvscuserfont{\the\glsshorttok}}
{\the\glslabeltok}
```

```
\glsxtrshortlonguserdescname
```

Expands to the value for the name key for styles like short-long-user-desc. The default definition is:

```
\protect\glsabbrvuserfont{\the\glsshorttok}%
\protect\glsxtruserparen
  {\protect\glslonguserfont{\the\glslongpltok}}%
  {\the\glslabeltok}
```

```
\glsxtruserlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}
{⟨short-fmt-cs⟩}
```

This command is used on the first use of \gls or with \glsxtrfull by styles like long-short-user to format the long form followed by the short form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtruserlongshortformat}[4]{%
  \glsxtrlongformat{#1}{#2}{#3}%
  \glsxtrusershortformat{#1}{#4}%
}
```

\Glsxtruserlongshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for sentence case.

\GLSxtruserlongshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for all caps.

\glsxtruserlongshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

This command is used on the first use of \glspl or with \glsxtrfullpl by styles like long-short-user to format the plural long form followed by the plural short form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtruserlongshortplformat}[4]{%
  \glsxtrlongplformat{#1}{#2}{#3}%
  \glsxtrusershortplformat{#1}{#4}%
}
```

\Glsxtruserlongshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for sentence case.

\GLSxtruserlongshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for all caps.

\glsxtrusershortlongformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

This command is used on the first use of \gls or with \glsxtrfull by styles like short-long-user to format the short form followed by the long form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtrusershortlongformat}[4]{%
  \glsxtrshortformat{#1}{#2}{#3}%
  \glsxtruserlongformat{#1}{#4}%
}
```

\Glsxtrusershortlongformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for sentence case.

\GLSxtrusershortlongformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

As above but for all caps.

\glsxtrusershortlongplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

This command is used on the first use of \glspl or with \glsxtrfullpl by styles like short-long-user to format the plural short form followed by the plural long form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtrusershortlongplformat}[4]{%
  \glsxtrshortplformat{#1}{#2}{#3}%
  \glsxtruserlongplformat{#1}{#4}%
}
```

\Glsxtrusershortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for sentence case.

\GLSxtrusershortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for all caps.

\glsxtrusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used to format the singular short form in parentheses (with \glsxtruserparen) on the first use of \gls or \Gls or with \glsxtrfull or \Glsxtrfull for styles like long-short-user. The default definition is:

```
\newcommand*{\glsxtrusershortformat}[2]{%
 \glsxtruserparen
  {\glsxtrshortformat{#1}{}{#2}}%
  {#1}%
}
```

\glsxtrusershortplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortformat but for the first use of \glspl or with \glsxtrfull for styles like long-short-user. This has a similar definition to the above but with \glsxtrshortplformat.

\GLSxtrusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortformat but is used with the all caps \GLS or \GLSxtrfull. This uses \GLSxtruserparen instead of \glsxtruserparen.

\GLSxtrusershortplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortplformat but is used with the all caps \GLSpl or \GLSxtrfullpl. This uses \GLSxtruserparen instead of \glsxtruserparen.

\glsxtrpostusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used in the post-link hook to format the short form in parentheses for styles like long−postshort−user. The default definition is:

```
\newcommand*{\glsxtrpostusershortformat}[2]{%
  \glsxtrifallcaps
  {\GLSxtrusershortformat{#1}{#2}}%
  {\glsxtrusershortformat{#1}{#2}}%
}
```

Note that this doesn't check if the plural form was used. If you require this, you will need to redefined this command to include \glsifplural:

```
\renewcommand*{\glsxtrpostusershortformat}[2]{%
  \glsifplural
  {%
    \glsxtrifallcaps
    {\GLSxtrusershortplformat{#1}{#2}}%
    {\glsxtrusershortplformat{#1}{#2}}%
  }%
  {%
    \glsxtrifallcaps
    {\GLSxtrusershortformat{#1}{#2}}%
    {\glsxtrusershortformat{#1}{#2}}%
  }%
}
```

\glsxtruserlongformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used to format the singular long form in parentheses (with \glsxtruserparen) on the first use of \gls or \Gls or with \glsxtrfull for styles like short−long−user. The default definition is:

```
\newcommand*{\glsxtruserlongformat}[2]{%
 \glsxtruserparen{\glsxtrlongformat{#1}{}{#2}}{#1}%
}
```

\GLSxtruserlongformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtruserlongformat but all caps. This uses \GLSxtruserparen instead of \glsxtruserparen.

\glsxtruserlongplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtruserlongformat but for the first use of \glspl or with \glsxtrfull for styles like short−long−user. This has a similar definition to \glsxtruserlongformat but with \glsxtrlongplformat.

\GLSxtruserlongplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtruserlongplformat but all caps. This uses \GLSxtruserparen instead of \glsxtruserparen.

\glsxtrpostuserlongformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used in the post-link hook to format the long form in parentheses for styles like short−postlong−user. The default definition is:

```
\newcommand*{\glsxtrpostuserlongformat}[2]{%
  \glsxtrifallcaps
  {\GLSxtruserlongformat{#1}{#2}}%
  {\glsxtruserlongformat{#1}{#2}}%
}
```

Note that, as with \glsxtrpostusershortformat, this doesn't check if the plural form was used. If you require this, you will need to redefined this command to include \glsifplural.

4.5.1.3.4. Footnote Styles

These commands are only used by the footnote styles.

\glsxtrfootnotename

This command should expand to the value of the name key. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
```

```
\glsxtrfootnotedescname
```

This command should expand to the value of the `name` key for styles like footnote-desc. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}%
\protect\glsxtrfullsep{\the\glslabeltok}%
\protect\glsxtrparen
 {\glsxplongfont{\the\glslongtok}{\glscategorylabel}}%
```

```
\glsxtrfootnotedescsort
```

This command should expand to the value of the `sort` key for styles like footnote-desc. The default definition is simply `\the\glsshorttok`.

This command is irrelevant with the "unsrt" family of commands.

```
\glslongfootnotefont{⟨text⟩}
```

The formatting command used for the long form in the footnote styles. The default is to simply use `\glslongdefaultfont`.

```
\glsfirstlongfootnotefont{⟨text⟩}
```

The formatting command used for the first use long form in the footnote styles. The default is to simply use `\glslongfootnotefont`.

```
\glsxtrabbrvfootnote{⟨entry-label⟩}{⟨text⟩}
```

The command that produces the footnote. The default definition ignores the first argument:

```
\newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}
```

```
\glsxtrfootnotelongformat{⟨entry-label⟩}{⟨fmt-cs⟩}
```

This command is used within the footnote to display the long form formatted with ⟨*fmt-cs*⟩ for the footnote styles on first use of `\gls`, `\Gls` and `\GLS`. The default definition is simply:

```
\newcommand*{\glsxtrfootnotelongformat}[2]{%
  \glsxtrlongformat{#1}{}{#2}%
}
```

For example, if the footnote should start with an uppercase letter then simply redefine this to use `\Glsxtrlongformat` instead:

```
\renewcommand*{\glsxtrfootnotelongformat}[2]{%
  \Glsxtrlongformat{#1}{}{#2}%
}
```

```
\glsxtrfootnotelongplformat{⟨entry-label⟩}{⟨fmt-cs⟩}
```

This command is used within the footnote to display the plural long form formatted with ⟨*fmt-cs*⟩ for the footnote styles on first use of `\glspl`, `\Glspl` and `\GLSpl`. The default definition is simply:

```
\newcommand*{\glsxtrfootnotelongplformat}[2]{%
  \glsxtrlongplformat{#1}{}{#2}%
}
```

```
\glsxtrpostfootnotelongformat{⟨entry-label⟩}{⟨fmt-cs⟩}
```

This command is used for the "postfootnote" styles. This is simply defined to do `\glsxtrfootnotelongformat`. Note that there's no plural equivalent as the "postfootnote" styles don't check if the plural command (`\glspl` etc) was used.

4.5.1.3.5. No-Long Styles

These commands are used by the "nolong" styles.

```
\glsxtrshortnolongname
```

This command should expand to the value of the name key for styles like short-nolong. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
```

```
\glsxtrshortdescname
```

This command should expand to the value of the name key for styles like short-nolong-desc. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}%
\protect\glsxtrfullsep{\the\glslabeltok}%
\protect\glsxtrparen
  {\glsxplongfont{\the\glslongtok}{\glscategorylabel}}%
```

4.5.1.3.6. No-Short Styles

These commands are used by the "noshort" styles.

```
\glsxtrlongnoshortdescname
```

This command should expand to the value of the name key for styles like long-noshort-desc. The default definition is:

```
\glsxplongfont{\the\glslongtok}{\glscategorylabel}
```

```
\glsxtrlongnoshortname
```

This command should expand to the value of the name key for styles like long-noshort. The default definition is:

```
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
```

4.5.1.3.7. Hyphen Styles

These are commands used by the "hyphen" styles. They are designed to work with the markwords and markshortwords attributes.

\glsabbrvhyphenfont{⟨text⟩}

The formatting command used for the short form in the hyphen styles. The default is to simply use \glsabbrvdefaultfont.

\glsfirstabbrvhyphenfont{⟨text⟩}

The formatting command used for the short form in the hyphen styles on first use. The default is to simply use \glsabbrvhyphenfont.

\glslonghyphenfont{⟨text⟩}

The formatting command used for the long form in the hyphen styles. The default is to simply use \glslongdefaultfont.

\glsfirstlonghyphenfont{⟨text⟩}

The formatting command used for the long form in the hyphen styles on first use. The default is to simply use \glslonghyphenfont.

\glsxtrhyphensuffix *initial:* \glsxtrabbrvpluralsuffix

Short plural suffix used by the "hyphen" styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsxtrlonghyphenshortsort

Expands to the sort value for the styles like long-hyphen-short-hyphen. This defaults to the original short value (\glsxtrorgshort). This command is irrelevant with the "unsrt" family of commands.

\glsxtrshorthyphenlongsort

Expands to the sort value for the styles like short-hyphen-long-hyphen. This defaults to the original short value (\glsxtrorgshort). This command is irrelevant with the "unsrt" family of commands.

\glsxtrlonghyphennoshortsort

Expands to the sort value for the styles like long-hyphen-noshort-noreg. This defaults to the original short value (\glsxtrorgshort). This command is irrelevant with the "unsrt" family of commands.

\glsxtrlonghyphennoshortdescsort

Expands to the sort value for the styles like long-hyphen-noshort-desc-noreg. This defaults to the original long value (\glsxtrorglong). This command is irrelevant with the "unsrt" family of commands.

\glsxtrlonghyphenshort{⟨entry-label⟩}{⟨long⟩}{⟨short⟩}{⟨insert⟩}

Formats the long and short form for the full or first use long-hyphen-short-hyphen style. This uses \glsxtrifhyphenstart to test if the ⟨*insert*⟩ starts with a hyphen. If it does, \glsxtrwordsep is locally set to \glsxtrwordsephyphen to replace the inter-word spaces with hyphens. The short form is placed in parentheses with \glsxtrparen, preceded by the \glsxtrfullsep separator. The ⟨*insert*⟩ is placed after both the long and the short form.

\GLSxtrlonghyphenshort{⟨entry-label⟩}{⟨long⟩}{⟨short⟩}{⟨insert⟩}

As above, but the ⟨*insert*⟩ is converted to all caps. The ⟨*short*⟩ and ⟨*long*⟩ arguments should be supplied as all caps. Note that it's not possible to simply do \glsxtrlonghyphenshort with \MakeUppercase{⟨*insert*⟩} as the argument as this will interfere with the check to determine if ⟨*insert*⟩ starts with a hyphen.

\glsxtrlonghyphennoshort{⟨entry-label⟩}{⟨long⟩}{⟨insert⟩}

Formats the long form for the full or first use long-hyphen-noshort-desc-noreg style. This uses \glsxtrifhyphenstart to test if the ⟨*insert*⟩ starts with a hyphen. If it does, \glsxtrwordsep is locally set to \glsxtrwordsephyphen to replace the inter-word spaces with hyphens. The ⟨*insert*⟩ is placed after the long form.

\GLSxtrlonghyphennoshort{⟨entry-label⟩}{⟨long⟩}{⟨insert⟩}

As above but converts ⟨*insert*⟩ to all caps. The ⟨*long*⟩ argument should already be in all caps. Note that it's not possible to simply do \glsxtrlonghyphennoshort with \MakeUppercase{⟨*insert*⟩} as the argument as this will interfere with the check to determine if ⟨*insert*⟩ starts with a hyphen.

\glsxtrlonghyphen{⟨*entry-label*⟩}{⟨*long*⟩}{⟨*insert*⟩}

Formats the long form for the full or first use long-hyphen-postshort-hyphen style. This is similar to the above, but the ⟨*insert*⟩ argument is only used to check if it starts with a hyphen. The actual ⟨*insert*⟩ is placed in the post-link hook.

\xpglsxtrposthyphenshort

This command is used in the post-link hook for the long-hyphen-postshort-hyphen style on first use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphenshort for all caps or \glsxtrposthyphenshort otherwise. Note that this doesn't show the plural by default. If you require the plural form, you need to redefine this to add a check with \glsifplural:

```
\renewcommand*{\xpglsxtrposthyphenshort}{%
  \glsifplural
  {%
    \glsxtrifallcaps
    {%
      \expandafter\GLSxtrposthyphenshortpl
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
    {%
      \expandafter\glsxtrposthyphenshortpl
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
  }%
  {%
    \glsxtrifallcaps
    {%
      \expandafter\GLSxtrposthyphenshort
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
    {%
      \expandafter\glsxtrposthyphenshort
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
  }%
```

```
}
```

\glsxtrposthyphenshort{⟨*entry-label*⟩}{⟨*insert*⟩}

If ⟨*insert*⟩ starts with a hyphen, \glsxtrwordsep is locally set to \glsxtrwordsep-hyphen to replace the inter-word spaces with hyphens. The ⟨*insert*⟩ encapsulated with \glsfirstlonghyphenfont is then done (to complete the long form, which has already been displayed with \glsxtrlonghyphen in the link text). Then the short form followed by the ⟨*insert*⟩ is placed in parentheses (with \glsxtrparen preceded by \glsxtrfullsep).

\GLSxtrposthyphenshort{⟨*entry-label*⟩}{⟨*insert*⟩}

As above but all caps.

\glsxtrposthyphenshortpl{⟨*entry-label*⟩}{⟨*insert*⟩}

As \glsxtrposthyphenshort but plural.

\GLSxtrposthyphenshortpl{⟨*entry-label*⟩}{⟨*insert*⟩}

As above but all caps.

\xpglsxtrposthyphensubsequent

This command is used in the post-link hook for the long-hyphen-postshort-hyphen style on subsequent use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphensubsequent for all caps or \glsxtrposthyphen-subsequent otherwise.

\glsxtrposthyphensubsequent{⟨*entry-label*⟩}{⟨*insert*⟩}

This command is used in the post-link hook for the long-hyphen-postshort-hyphen style on subsequent use. Only the ⟨*insert*⟩ is done.

\GLSxtrposthyphensubsequent{⟨*entry-label*⟩}{⟨*insert*⟩}

As above but all caps.

\glsxtrshorthyphenlong{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩}

Formats the short and long form for the full or first use short−hyphen−long−hyphen style. Similar to \glsxtrlonghyphenshort but the short and long forms are swapped.

\GLSxtrshorthyphenlong{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩}

As above, but the ⟨*insert*⟩ is converted to all caps. The ⟨*short*⟩ and ⟨*long*⟩ arguments should be supplied as all caps. Note that it's not possible to simply do \glsxtrshorthyphenlong with \MakeUppercase{⟨*insert*⟩} as the argument as this will interfere with the check to determine if ⟨*insert*⟩ starts with a hyphen.

\glsxtrshorthyphen{⟨short⟩}{⟨entry-label⟩}{⟨insert⟩}

Formats the short form for the full or first use short−hyphen−postlong−hyphen style. The ⟨*insert*⟩ argument is only used to check if it starts with a hyphen. The actual ⟨*insert*⟩ is placed in the post-link hook.

\xpglsxtrposthyphenlong

This command is used in the post-link hook for the short−hyphen−postlong−hyphen style on first use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphenlong for all caps or \glsxtrposthyphenlong otherwise. Note that this doesn't show the plural by default. If you require the plural form, you need to redefine this to add a check with \glsifplural:

```
\renewcommand*{\xpglsxtrposthyphenlong}{%
  \glsifplural
  {%
    \glsxtrifallcaps
    {%
      \expandafter\GLSxtrposthyphenlongpl
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
    {%
      \expandafter\glsxtrposthyphenlongpl
         \expandafter\glslabel
         \expandafter{\glsinsert}%
    }%
```

```
    }%
    {%
      \glsxtrifallcaps
      {%
        \expandafter\GLSxtrposthyphenlong
           \expandafter\glslabel
           \expandafter{\glsinsert}%
      }%
      {%
        \expandafter\glsxtrposthyphenlong
           \expandafter\glslabel
           \expandafter{\glsinsert}%
      }%
    }%
  }
```

\glsxtrposthyphenlong{⟨*entry-label*⟩}{⟨*insert*⟩}

This command is used in the post-link hook for the short−hyphen−postlong−hyphen style on first use. Similar to \glsxtrposthyphenshort but shows the long form instead of the short form.

\GLSxtrposthyphenlong{⟨*entry-label*⟩}{⟨*insert*⟩}

As above but all caps.

\glsxtrposthyphenlongpl{⟨*entry-label*⟩}{⟨*insert*⟩}

As \glsxtrposthyphenlong but plural.

\GLSxtrposthyphenlongpl{⟨*entry-label*⟩}{⟨*insert*⟩}

As above but all caps.

4.5.1.3.8. Only Styles

These are commands used by the “only” styles, such as long−only−short−only.

\glsabbrvonlyfont{⟨*text*⟩}

The formatting command used for the short form in the only styles. The default is to simply use \glsabbrvdefaultfont.

\glsfirstabbrvonlyfont{⟨text⟩}

The formatting command used for the short form in the only styles on first use. The default is to simply use \glsabbrvonlyfont.

\glslongonlyfont{⟨text⟩}

The formatting command used for the long form in the only styles. The default is to simply use \glslongdefaultfont.

\glsfirstlongonlyfont{⟨text⟩}

The formatting command used for the long form in the only styles on first use. The default is to simply use \glslongonlyfont.

\glsxtronlysuffix *initial:* \glsxtrabbrvpluralsuffix

Short plural suffix used by the "only" styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsabbrvsconlyfont{⟨text⟩}

The formatting command used for the short form in the "sc-only" styles. The default is to simply use \glsabbrvscfont.

\glsfirstabbrvsconlyfont{⟨text⟩}

The formatting command used for the short form in the "sc-only" styles on first use. The default is to simply use \glsabbrvsconlyfont.

\glsxtrsconlyrevert{⟨text⟩}

Counteracts the effect of \glsabbrvsconlyfont. The default is \glsxtrscrevert. If you redefine \glsabbrvsconlyfont, you will need to redefine \glsxtrsconlyrevert as applicable.

\glsxtrsconlysuffix *initial:* \glsxtrscsuffix

Short plural suffix used by the "sc-only" styles. This defaults to \glsxtrscsuffix.

```
\glsxtronlyname
```

Expands to the value for the `name` key for the “only” styles. The default definition is:

```
\protect\glsabbrvonlyfont{\the\glsshorttok}
```

```
\glsxtronlydescname
```

Expands to the value for the `name` key for the “only” styles where the description should be described, such as long-only-short-only-desc. The default definition is:

```
\protect\glslongfont{\the\glslongtok}
```

```
\glsxtronlydescsort
```

Expands to the value for the `sort` key for the “only” styles where the description should be described, such as long-only-short-only-desc. The default definition is `\the\glslongtok`.

This command is irrelevant with the “unsrt” family of commands.

```
\glsxtrsconlyname
```

Expands to the value for the `name` key for the “sc-only” styles. The default definition is:

```
\protect\glsabbrvsconlyfont{\the\glsshorttok}
```

```
\glsxtrsconlydescname
```

Expands to the value for the `name` key for the “sc-only” styles where the description should be described. The default definition is to simply use `\glsxtronlydescname`.

\glsxtrsconlydescsort

Expands to the value for the sort key for the "sc-only" styles where the description should be described, such as long-only-short-only-desc. The default definition is to simply use \glsxtronlydescsort.

This command is irrelevant with the "unsrt" family of commands.

4.5.1.3.9. Fonts

These are commands used by styles that use a particular font shape or size, identified by one of the following two-letter tags: "sc" (\textsc), "sm" (\textsmaller) or "em" (\emph).

For the "sc-user" styles, see §4.5.1.3.3. For the "sc-only" styles, see §4.5.1.3.8.

\glsabbrvscfont{⟨*text*⟩}

Formatting for the "sc" short form. This uses \textsc to apply a small caps style, so your document font needs to support it.

\textsc uses small capital glyphs for lowercase characters. Uppercase characters show as normal capitals. This means that you need to use lowercase characters in the abbreviation.

\glsfirstabbrvscfont{⟨*text*⟩}

Formatting for the "sc" short form shown on first use. This defaults to \glsabbrvscfont.

\glsxtrscrevert{⟨*text*⟩}

Counteracts the effect of \glsabbrvscfont. This defaults to \glstextup. If you redefine \glsabbrvscfont, you will need to redefine \glsxtrscrevert as applicable.

\glsxtrscsuffix

Short plural suffix used by the “sc” styles. This needs to counteract the smallcaps, so it’s defined as:

```
\protect\glstextup\glsxtrabbrvpluralsuffix
```

\glsabbrvsmfont{⟨*text*⟩}

Formatting for the “sm” short form. This uses \textsmaller, which is defined by the relsize package. You will need to load that package if you want to use any of the “sm” styles.

> \textsmaller reduces the font size, so if you want to use it to simulate small caps, you need to use uppercase characters in the abbreviation.

\glsfirstabbrvsmfont{⟨*text*⟩}

Formatting for the “sm” short form shown on first use. This defaults to \glsabbrvsmfont.

\glsxtrsmrevert{⟨*text*⟩}

Counteracts the effect of \glsabbrvsmfont. This defaults to \textlarger. If you redefine \glsabbrvsmfont, you will need to redefine \glsxtrsmrevert as applicable.

\glsxtrsmsuffix *initial:* \glsxtrabbrvpluralsuffix

Short plural suffix used by the “sm” styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsabbrvemfont{⟨*text*⟩}

Formatting for the “em” short form. This uses \emph.

\glsfirstabbrvemfont{⟨*text*⟩}

Formatting for the “em” short form shown on first use. This defaults to \glsabbrvemfont.

\glsxtremrevert{⟨text⟩}

Counteracts the effect of \glsabbrvemfont. This defaults to \textup. If you redefine \glsabbrvemfont, you will need to redefine \glsxtremrevert as applicable.

\glsxtremsuffix *initial:* \glsxtrabbrvpluralsuffix

Short plural suffix used by the "em" styles. This defaults to \glsxtrabbrvpluralsuffix.

\glslongemfont{⟨text⟩}

Formatting for the "em" long form. This uses \emph.

\glsfirstlongemfont{⟨text⟩}

Formatting for the "em" short form shown on first use. This defaults to \glslongemfont.

4.5.2. Advanced Style Commands

These commands should typically not be needed in a document, but are provided for advanced users. See §4.5.1.3 for commands to adjust the predefined abbreviation styles.

\glssetabbrvfmt{⟨category⟩}

Sets the current formatting commands (§4.5.3.2) associated with the abbreviation style associated with the given category. That is, the command redefinitions provided in the third argument (⟨*display definitions*⟩) of \newabbreviationstyle are applied.

If no abbreviation style has been set for the given category, the style associated with the abbreviation category is used.

This command is used:

- At the start of \glsentryfmt if the current entry has the short field set. This ensures that the \gls-like commands use the appropriate formatting.
- At the start of \glsxtrassignfieldfont if the current entry has the short field set. This ensures that the \gls-like commands use the appropriate formatting (where possible).
- At the start of \glsxtrshort, \glsxtrlong, \glsxtrfull and their plural and case-changing variants.

- At the start of \glossentryname, \glossentrynameother, \glossentrydesc, \glossentrysymbol, and their sentence-case variants.

\glsuseabbrvfont{⟨style-name⟩}{⟨text⟩}

A robust command that applies the abbreviation font for the given category to the supplied text.

\glsuselongfont{⟨style-name⟩}{⟨text⟩}

A robust command that applies the long font for the given category to the supplied text.

\GlsXtrUseAbbrStyleSetup{⟨style-name⟩}

This implements the given abbreviation style's setup code. Note that this expects the placeholder macros and token registers to be set. This may be used in the ⟨*setup*⟩ of \newabbreviationstyle to inherit the setup code of a related style.

\GlsXtrUseAbbrStyleFmts{⟨style-name⟩}

This implements the given abbreviation style's display definitions code. This may be used in the ⟨*display definitions*⟩ of \newabbreviationstyle to inherit the formatting of a related style.

\xpglsxtrpostabbrvfootnote

This is used by styles like postfootnote to ensure that the label and inner and outer formatting are expanded before being passed to \glsxtrpostabbrvfootnote, otherwise they may lose their definitions before the footnote text is typeset.

\glsxtrpostabbrvfootnote{⟨entry-label⟩}{⟨fmt-code⟩}

This is used by the footnote styles that defer the footnote to the post-link hook. The default definition is:

```
\newrobustcmd*{\glsxtrpostabbrvfootnote}[2]{%
 \glsxtrabbrvfootnote{#1}%
 {#2\glsxtrpostfootnotelongformat
   {#1}{\glsfirstlongfootnotefont}}%
}
```

The second argument will be the expansion of `\glsxtrassignlinktextfmt`, to allow the inner formatting to be picked up, if required.

`\glsxtrifhyphenstart{`⟨*text*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This command is used by the hyphen styles to determine if the insert material starts with a hyphen. Does ⟨*true*⟩ if ⟨*text*⟩ starts with a hyphen otherwise does ⟨*false*⟩.

`\GlsXtrWarnDeprecatedAbbrStyle{`⟨*old-name*⟩`}{`⟨*new-name*⟩`}`

This command is used to generate a warning (with `\GlossariesExtraWarning`) if a deprecated abbreviation style is used.

4.5.3. Defining New Abbreviation Styles

If none of the predefined styles suit your requirements, you can define your own custom style using:

`\newabbreviationstyle{`⟨*style-name*⟩`}{`⟨*setup*⟩`}{`⟨*display definitions*⟩`}`

The first argument is the style name. This is used internally to form control sequences, so the name shouldn't contain any special characters.

The second argument sets up the information that's required when an abbreviation is defined (which is why the style must be set before the abbreviations with that style are defined). The relevant commands for this argument are listed in §4.5.3.1.

The third argument defines the commands that determine how the display style (`\gls`) and the inline style (`\glsxtrfull`) are formatted. The relevant commands for this argument are listed in §4.5.3.2.

`\renewabbreviationstyle{`⟨*style-name*⟩`}{`⟨*setup*⟩`}{`⟨*display definitions*⟩`}`

Redefines an existing abbreviation style.

`\letabbreviationstyle{`⟨*new style*⟩`}{`⟨*existing style*⟩`}`

Defines a synonym of an existing abbreviation style.

4.5.3.1. Style Initialisation Hooks

The style initialisation hooks should be placed in the second argument (⟨*setup*⟩) of `\newabbreviationstyle`. They ensure that all the fields are correctly initialised when the entry is defined with the underlying `\newglossaryentry` command. They may also be used to set category attributes.

The following is prepended to ⟨*setup*⟩ to initialise the final hook:

```
\renewcommand*{\GlsXtrPostNewAbbreviation}{}
```

When an entry is defined with `\newabbreviation`, the following steps are performed:

1. Token registers are initialised to the information provided in the arguments of `\newabbreviation`: `\glskeylisttok`, `\glslabeltok`, `\glsshorttok` and `\glslongtok`.

2. The commands `\glsxtrorgkeylist`, `\glsxtrorgshort` and `\glsxtrorglong` are defined to the options, short and long values supplied to `\newabbreviation`. (The `\glskeylisttok` `\glsshorttok` and `\glslongtok` token registers may be changed before the entry is actually defined. These commands may be used to obtain the original values.)

3. `\ExtraCustomAbbreviationFields` is initialised to do nothing.

4. Accessibility settings are initialised, if required. These redefine `\ExtraCustomAbbreviationFields` to set the accessibility fields.

5. The command `\glscategorylabel` is defined to `abbreviation`.

6. The options list is parsed for the following keys: `category` and, if accessibility is enabled, `access`, `textaccess`, `pluralaccess`, `firstaccess`, `firstpluralaccess`, `shortaccess`, `shortpluralaccess`, `longaccess`, and `longpluralaccess`.

7. The abbreviation style is applied for the category given by `\glscategorylabel` (which may have been changed when the options were parsed in the previous step) or the fallback if no abbreviation style is associated with that category. This performs both the ⟨*setup*⟩ and ⟨*display definitions*⟩ provided when the style was defined with `\newabbreviationstyle`.

8. The long plural form is initialised to its default value (⟨*long*⟩`\glspluralsuffix`).

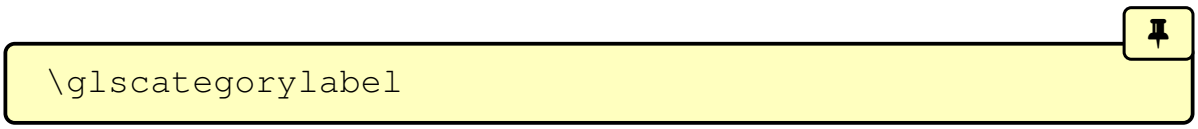
9. The `markwords` attribute, if set, is implemented for the singular long form. It will also mark the entry as having a description with formatting (using `\glsexclapplyinnerfmtfield`).

10. The markshortwords attribute is implemented, if set, otherwise the insertdots attribute is implemented, if set, for the singular short form.
11. The aposplural attribute is implemented, if set, otherwise the noshortplural attribute is implemented, if set. This step will set the default short plural.
12. \glsshorttok is updated to reflect any changes.
13. The \glsxtrnewabbrevpresetkeyhook hook is performed.
14. The options list is parsed for the shortplural and longplural keys. The \glskeylisttok token is updated to only include the remaining keys that haven't yet been processed.
15. The markwords attribute, if set, is implemented for the plural long form.
16. The markshortwords attribute, if set, otherwise the insertdots attribute, if set, is implemented for the plural short form.
17. The \glsshortpltok and \glslongpltok registers are set.
18. \newabbreviationhook performed.
19. The entry is defined using \newglossaryentry with the key value list:

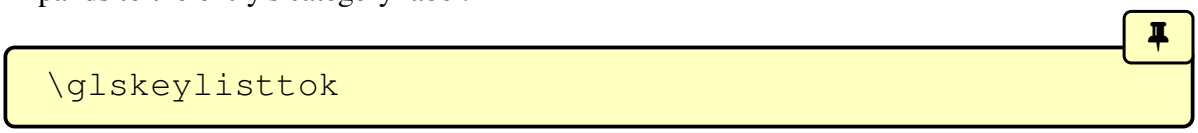
```
type={\glsxtrabbrvtype},
category={abbreviation},
short={\the\glsshorttok},
shortplural={\the\glsshortpltok},
long={\the\glslongtok},
longplural={\the\glslongpltok},
name={\the\glsshorttok},
\CustomAbbreviationFields,
\ExtraCustomAbbreviationFields
\the\glskeylisttok
```

20. Add the name, first, firstplural, text and plural keys to the list of inner formatting exclusions, as they include formatting commands.
21. Final hook \GlsXtrPostNewAbbreviation performed.

Note that when these hooks (except the last) are used, the entry hasn't yet been defined. However, some information will have already been picked up from the arguments of \newabbreviation. These can be accessed in the hooks using the following (but make sure they are fully expanded):

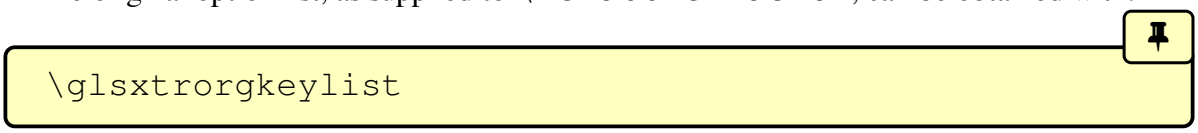


Expands to the entry's category label.

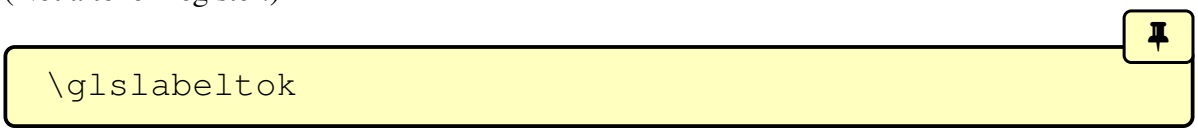


A token register that contains the options that were passed to \newabbreviation with pre-processed options removed. Use \the\glskeylisttok to expand it.

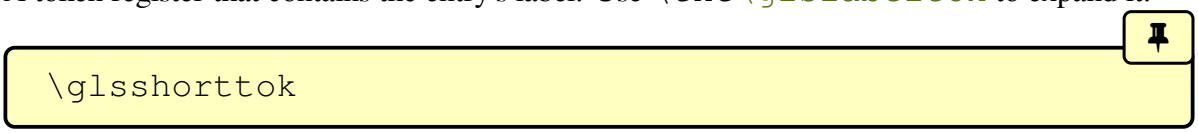
The original option list, as supplied to \newabbreviation, can be obtained with:



(Not a token register.)

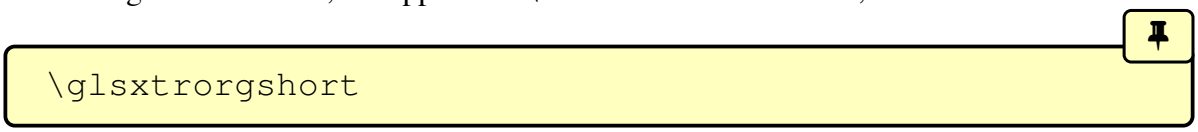


A token register that contains the entry's label. Use \the\glslabeltok to expand it.

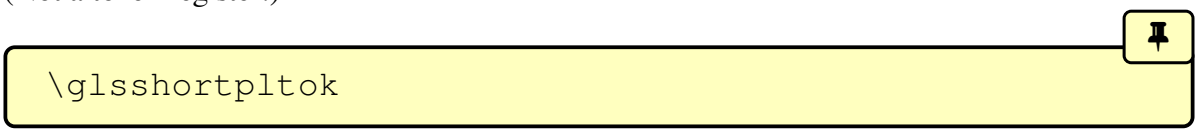


A token register that contains the short form (which may have been modified after being passed to \newabbreviation). Use \the\glsshorttok to expand it.

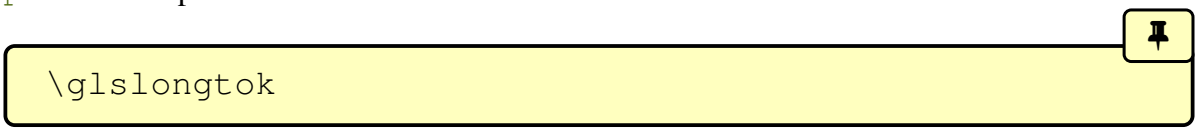
The original short form, as supplied to \newabbreviation, can be obtained with:



(Not a token register.)



A token register that contains the short plural form (which may have been obtained from the short form or modified after being passed to \newabbreviation). Use \the\glsshortpltok to expand it.



A token register that contains the long form (which may have been modified after being passed to \newabbreviation). Use \the\glslongtok to expand it.

The original long form, as supplied to \newabbreviation, can be obtained with:

\glsxtrorglong

(Not a token register.)

\glslongpltok

A token register that contains the long plural form (which may have been obtained from the long form or modified after being passed to \newabbreviation). Use \the\glslongpltok to expand it.

\ExtraCustomAbbreviationFields

Expands to additional field definitions for the entry. This is used to add the accessibility fields (such as shortaccess), if enabled. The abbreviation style may append (\appto) or prepend (\preto) additional information, if required, to this hook.

If you alter this hook, make sure that you include the trailing comma after each ⟨*key*⟩={value}, including the last one.

\CustomAbbreviationFields

Expands to the default field definitions for the entry. Take care to protect any commands that shouldn't be expanded. The comma may be omitted from the final ⟨*key*⟩={value}.

\GlsXtrPostNewAbbreviation

A hook that's used after the entry has been defined (at the end of \newabbreviation). This can be used to set category attributes, define the post-link hook, or mark the entry as having a complex style (with \glsxtrsetcomplexstyle).

For example, the long–short abbreviation style includes the following in ⟨*setup*⟩:

```
\renewcommand*{\GlsXtrPostNewAbbreviation}{%
  \glsxtrsetcomplexstyle{\the\glslabeltok}{3}%
  \glshasattribute{\the\glslabeltok}{regular}%
  {%
    \glssetattribute
      {\the\glslabeltok}{regular}{false}%
```

```
    }%
    {}%
}
```

Note that in the above, the commands within the definition of `\GlsXtrPostNewAbbreviation` are all expanded when that hook is used. However, if this hook defines other commands or hooks that will be used later, then make sure that the definitions of those commands use the inner hook's own placeholder commands.

Remember that the post-link hook uses `\glslabel` to reference the current label. Don't use `\glslabeltok` as that will contain the label of the last abbreviation to be defined.

For example, the long-hyphen-postshort-hyphen style has:

```
\renewcommand*{\GlsXtrPostNewAbbreviation}{%
  \glsexclapplyinnerfmtfield{\the\glslabeltok}{desc}%
  \csdef{glsxtrpostlink\glscategorylabel}{%
    \glsxtrifwasfirstuse
    {%
      \expandafter\glsxtrposthyphenshort
       \expandafter\glslabel
       \expandafter{\glsinsert}%
    }%
    {%
      \expandafter\glsxtrposthyphensubsequent
       \expandafter\glslabel
       \expandafter{\glsinsert}%
    }%
  }%
  \glshasattribute{\the\glslabeltok}{regular}%
  {%
    \glssetattribute
      {\the\glslabeltok}{regular}{false}%
  }%
  {}%
}
```

In the above, `\glslabeltok` and `\glscategorylabel` are used in the parts that will be expanded at the end of `\newabbreviation`, but `\glslabel` and `\glsinsert` are used in the definition of the post-link hook, which won't be expanded until the entry is referenced in the document with a command such as `\gls`. (The use of `\expandafter` is included to assist `innertextformat`.)

```
\glsxtrsetcomplexstyle{⟨entry-label⟩}{⟨n⟩}
```

This command should go in the definition of `\GlsXtrPostNewAbbreviation` to indicate that the entry given by ⟨*entry-label*⟩ has an abbreviation style that is complex. The second argument ⟨*n*⟩ should be numeric and indicates why it doesn't work with `\glsfirst`, `\Glsfirst`, `\GLSfirst`, `\glsfirstplural`, `\Glsfirstplural` or `\GLSfirstplural`: 1 (all caps doesn't work), 2 (all caps and insert doesn't work), 3 (insert doesn't work).

```
\glsfirstinnerfmtabbrvfont{⟨text⟩}
```

This is a robust command that applies both `\glsfirstabbrvfont` and the inner formatting command `\glsxtrgenentrytextfmt`. This is used by the following command.

```
\glsfirstxpabbrvfont{⟨text⟩}{⟨category⟩}
```

If the `markshortwords` attribute is true, this does `\protect\glsfirstabbrvfont{⟨text⟩}` otherwise it does `\glsfirstinnerfmtabbrvfont{⟨text⟩}`.

This command is designed to be used within `\CustomAbbreviationFields` to set the `first` and `firstplural` keys, so it needs to partially expand within `\newabbreviation`. For example, the postfootnote includes the following lines in the definition of `\CustomAbbreviationFields`:

```
first={\glsfirstxpabbrvfont
  {\the\glsshorttok}{\glscategorylabel}},
firstplural={\glsfirstxpabbrvfont
  {\the\glsshortpltok}{\glscategorylabel}},
```

This will be expanded before being passed to `\newglossaryentry`. If the `markshortwords` attribute is true, this will end up as:

```
first={\protect\glsfirstabbrvfont{⟨short⟩}},
firstplural={\protect\glsfirstabbrvfont{⟨shortpl⟩}},
```

otherwise it will end up as:

```
first={\glsfirstinnerfmtabbrvfont{⟨short⟩}},
firstplural={\glsfirstinnerfmtabbrvfont{⟨shortpl⟩}},
```

where ⟨*short*⟩ and ⟨*shortpl*⟩ are, respectively, the values in the `\glsshorttok` and `\glsshortpltok` registers.

The placeholder registers and macros (such as `\glsshorttok` and `\glscategorylabel`) must be expanded before being passed to `\newglossaryentry` as their values are unreliable outside of `\newabbreviation`.

`\glsinnerfmtabbrvfont{`⟨*text*⟩`}`

This is a robust command that applies both `\glsabbrvfont` and the inner formatting command `\glsxtrgenentrytextfmt`. This is used by the following command.

`\glsxpabbrvfont{`⟨*text*⟩`}{`⟨*category*⟩`}`

If the `markshortwords` attribute is true, this does `\protect\glsabbrvfont{`⟨*text*⟩`}` otherwise it does `\glsinnerfmtabbrvfont{`⟨*text*⟩`}`. This command is designed for the `name`, `text` and `plural` keys within `\CustomAbbreviationFields`.

`\glsfirstinnerfmtlongfont{`⟨*text*⟩`}`

This is a robust command that applies both `\glsfirstlongfont` and the inner formatting command `\glsxtrgenentrytextfmt`. This is used by the following command.

`\glsfirstxplongfont{`⟨*category*⟩`}{`⟨*text*⟩`}`

If the `markwords` attribute is true, this does `\protect\glsfirstlongfont{`⟨*text*⟩`}` otherwise it does `\glsfirstinnerfmtlongfont{`⟨*text*⟩`}`. This command is designed for the `first` and `firstplural` keys within `\CustomAbbreviationFields`.

`\glsinnerfmtlongfont{`⟨*text*⟩`}`

This is a robust command that applies both `\glslongfont` and the inner formatting command `\glsxtrgenentrytextfmt`. This is used by the following command.

`\glsxplongfont{`⟨*text*⟩`}{`⟨*category*⟩`}`

If the `markwords` attribute is true, this does `\protect\glslongfont{`⟨*text*⟩`}` otherwise it does `\glsinnerfmtlongfont{`⟨*text*⟩`}`. This command is designed for the

name, text and plural keys within \CustomAbbreviationFields (if they should include the long form in their value, such as the long-noshort-desc style).

\glsxtrAccSuppAbbrSetNoLongAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the name, first, firstplural, text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The nameshortaccess, firstshortaccess and textshortaccess attributes are set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the name, first and text are just the formatted abbreviation. The first field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetNameLongAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the first, firstplural, text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The firstshortaccess and textshortaccess attributes are set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the first and text are just the formatted abbreviation. The name field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetFirstLongAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the name, text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The nameshortaccess and textshortaccess attributes are set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the name and text are just the formatted abbreviation. The first field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetTextShortAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The textshortaccess attribute is set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the text is just the formatted abbreviation. The name and first fields may just be the long form or may be a combination of short and long. The name may also be short but followed by the long form in the description.

\glsxtrAccSuppAbbrSetNameShortAttrs{⟨*category*⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the name field for the given category (using \glsxtrprovideaccsuppcmd). The nameshortaccess attribute is set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where only the name is just the formatted abbreviation. The first and text fields may just be the long form or may be a combination of short and long.

4.5.3.2. Style Formatting Commands

The final ⟨*display definitions*⟩ argument of \newabbreviationstyle should contain the redefinitions of the style commands listed here that are used to format abbreviations.

Whenever an abbreviation style is activated with commands like \setabbreviationstyle, \newabbreviation or \glssetabbrvfmt, ⟨*display definitions*⟩ are implemented.

If you simply want to adjust the formatting of one of the predefined styles, you should redefine the associated commands listed in §4.5.1.3.

The following initialisation is always prepended to ⟨*display definitions*⟩ so you can omit them if the default is appropriate for your style:

```
\renewcommand*{\glsxtrinlinefullformat}{%
 \glsxtrfullformat}%
\renewcommand*{\Glsxtrinlinefullformat}{%
 \Glsxtrfullformat}%
\renewcommand*{\GLSxtrinlinefullformat}{%
 \GLSxtrfullformat}%
\renewcommand*{\glsxtrinlinefullplformat}{%
 \glsxtrfullplformat}%
\renewcommand*{\Glsxtrinlinefullplformat}{%
 \Glsxtrfullplformat}%
\renewcommand*{\GLSxtrinlinefullplformat}{%
 \GLSxtrfullplformat}%
\let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
```

```
\let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
\let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
\let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
\let\GLSxtrsubsequentfmt\GLSxtrdefaultsubsequentfmt
\let\GLSxtrsubsequentplfmt\GLSxtrdefaultsubsequentplfmt
```

In the event that any styles omit defining the newer `\GLSxtrfullformat` or `\GLSxtrfullplformat`, these are also initialised to defaults but ideally they should have their definitions provided.

The minimal set of commands that should have their definitions provided are the abbreviation plural suffix (`\abbrvpluralsuffix`) the display full forms: `\glsxtrfullformat`, `\glsxtrfullplformat` and their case-changing variants.

The inline full form commands only need to be provided if they behave differently from the display full form. The subsequent use commands only need to be provided if the default (only show the short form) isn't suitable.

The content of ⟨*display definitions*⟩ is placed within the definition of an internal control sequence, so remember to use `##` instead of `#` to reference command parameters.

4.5.3.2.1. Suffix and Fonts

These are the generic suffix and font commands that vary according to the abbreviation style. The style should provide the appropriate definitions. The suffix should always be provided. The font commands are only required if the style applies any font formatting to either the long or short form.

`\abbrvpluralsuffix` *initial:* `\glsxtrabbrvpluralsuffix`

The plural suffix for the short form. For example, the long–short style defines this to just use `\glsxtrabbrvpluralsuffix`, but the smallcaps styles, such as long–short–sc define this to `\glsxtrscsuffix` in order to counteract the small caps font.

`\glsfirstabbrvfont{`⟨*text*⟩`}`

The font formatting command for the short form on first use. For example, the long–short–sc style has:

```
\renewcommand*\glsfirstabbrvfont[1]{%
 \glsfirstabbrvscfont{##1}}
```

```
\glsabbrvfont{⟨text⟩}
```

The font formatting command for the short form. For example, the long-short-sc style has:

```
\renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}
```

```
\glsxtrrevert{⟨text⟩}
```

This command is designed to counteract the effect of \glsabbrvfont if, for some reason, it shouldn't be applied to part of the abbreviation. For example, you may prefer not to have digits reduced with the smaller ("sm") styles.

```
\glsfirstlongfont{⟨text⟩}
```

The font formatting command for the long form on first use. For example, the long-short-sc style has:

```
\renewcommand*{\glsfirstlongfont}[1]{%
 \glsfirstlongdefaultfont{##1}}
```

```
\glslongfont{⟨text⟩}
```

The font formatting command for the long form. For example, the long-short-sc style has:

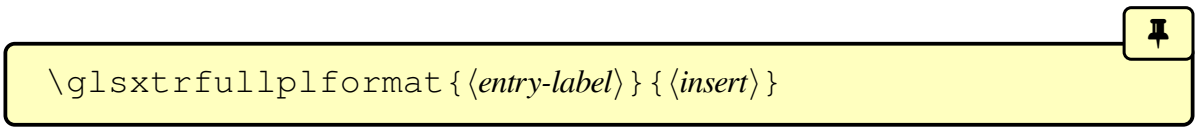
```
\renewcommand*{\glslongfont}[1]{%
 \glslongdefaultfont{##1}}
```

4.5.3.2.2. First Use Display Format

These commands always need to be provided.

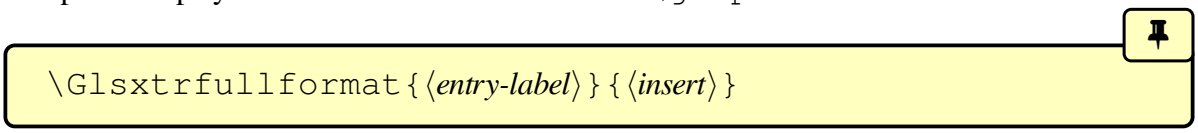
```
\glsxtrfullformat{⟨entry-label⟩}{⟨insert⟩}
```

The singular display full form used on the first use of \gls.



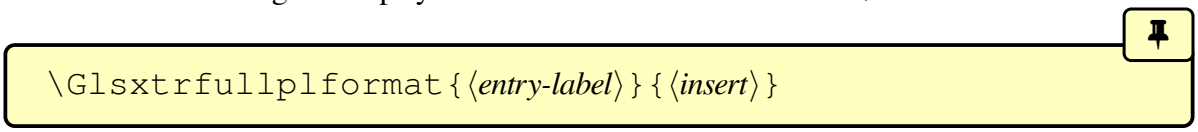
\glsxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}

The plural display full form used on the first use of \glspl.



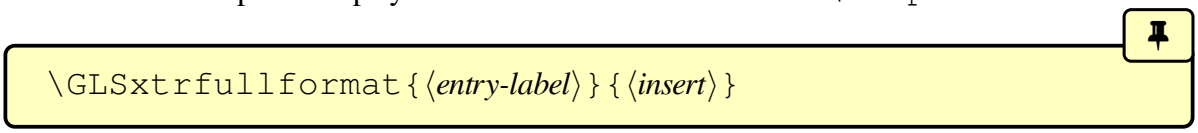
\Glsxtrfullformat{⟨entry-label⟩}{⟨insert⟩}

The sentence case singular display full form used on the first use of \Gls.



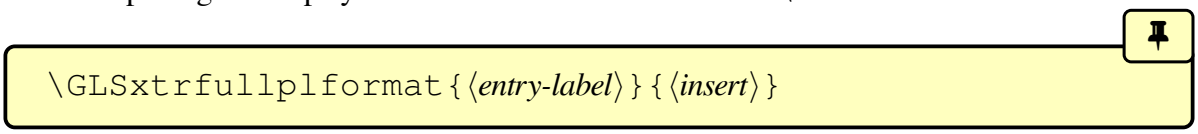
\Glsxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}

The sentence case plural display full form used on the first use of \Glspl.



\GLSxtrfullformat{⟨entry-label⟩}{⟨insert⟩}

The all caps singular display full form used on the first use of \GLS.

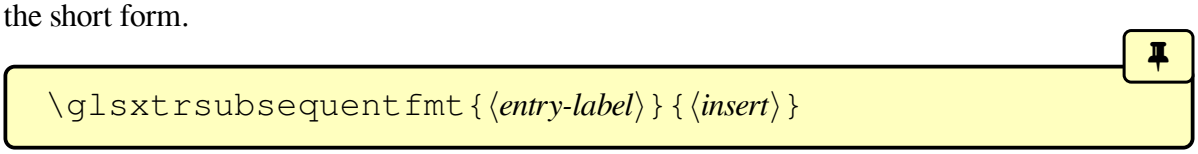


\GLSxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}

The all caps plural display full form used on the first use of \GLSpl.

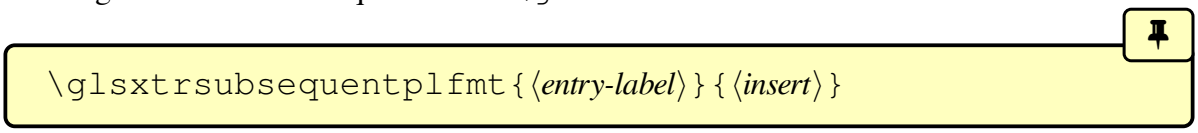
4.5.3.2.3. Subsequent Use Display Format

These commands only need to be provided if the \gls-like commands don't simply show the short form.



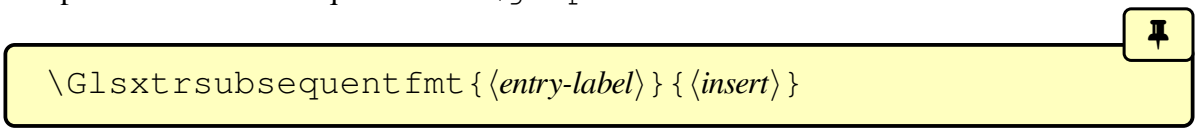
\glsxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The singular form for subsequent use of \gls.



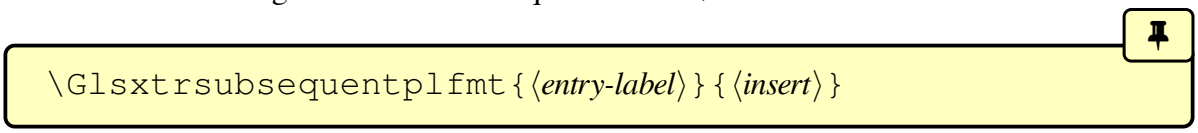
\glsxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The plural form for subsequent use of \glspl.



\Glsxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The sentence case singular form for subsequent use of \Gls.



\Glsxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The sentence case plural form for subsequent use of \Glspl.

\GLSxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The all caps singular form for subsequent use of \GLS.

\GLSxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The all caps plural form for subsequent use of \GLSpl.

The defaults all show the short form and insert encapsulated with the inner formatting \glsxtrgenentrytextfmt and \glsabbrvfont. The purpose of the inner formatting is to get it as close as possible to the actual text so \glsabbrvfont is placed outside of \glsxtrgenentrytextfmt.

The markshortwords attribute complicates matters as it inserts \glsxtrword and \glsxtrwordsep into the actual field value. In that case, the inner formatting is within \glsxtrword and \glsxtrwordsep, so only the insert material needs to be formatted.

If a custom style doesn't need to support innertextformat or \ifglsxtrinsertinside, it can reduce the complexity by omitting the inner formatting and conditionals, but this lack of support should be documented if the style is made generally available.

\glsxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default singular form for subsequent use of \gls.

\glsxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default plural form for subsequent use of \glspl.

\Glsxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default sentence case singular form for subsequent use of \Gls.

\Glsxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default sentence case plural form for subsequent use of \Glspl.

\GLSxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default all caps singular form for subsequent use of \GLS.

\GLSxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default all caps plural form for subsequent use of \GLSpl.

4.5.3.2.4. Inline Full Format

These commands only need to be provided if the inline full form is different from the display full form.

\glsxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}

The singular full form of \glsxtrfull.

\glsxtrinlinefullplformat{⟨entry-label⟩}{⟨insert⟩}

The plural full form of \glsxtrfullpl.

\Glsxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}

The sentence case singular full form of \Glsxtrfull.

\Glsxtrinlinefullplformat{⟨entry-label⟩}{⟨insert⟩}

The sentence case plural full form of \Glsxtrfullpl.

\GLSxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}

The all caps singular full form of \GLSxtrfull.

\GLSxtrinlinefullplformat{⟨entry-label⟩}{⟨insert⟩}

The all caps plural full form of \GLSxtrfullpl.

4.5.3.2.5. Wrapper Commands

These are commands that can be used in the definitions of the above to ensure that the appropriate accessibility fields and inner formatting is supported.

\glsxtrlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

This command is used in the definition of \glsxtrlong in some of the predefined abbreviation styles to format the long value of the entry identified by ⟨*entry-label*⟩ with the command ⟨*fmt-cs*⟩, which should take one argument.

Accessibility support is implemented with \glsaccesslong if the markwords attribute is true otherwise with \glsaccessfmtlong using \glsxtrgenentrytextfmt for the inner formatting.

This is then encapsulated (including or excluding the ⟨*insert*⟩, according to `\ifglsxtrinsertinside`) with ⟨*fmt-cs*⟩. If the ⟨*insert*⟩ content needs to be placed outside of ⟨*fmt-cs*⟩, it will be individually encapsulated with the inner formatting.

`\Glsxtrlongformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but sentence case.

`\GLSxtrlongformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but all caps.

`\glsxtrlongplformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As `\glsxtrlongformat`, but for the `longplural` field.

`\Glsxtrlongplformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but sentence case.

`\GLSxtrlongplformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but all caps.

`\glsxtrlongformatgrp{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As `\glsxtrlongformat`, but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

`\Glsxtrlongformatgrp{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but sentence case.

`\GLSxtrlongformatgrp{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As above, but all caps.

`\glsxtrlongplformatgrp{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

As \glsxtrlongplformat, but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\Glsxtrlongplformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but sentence case.

\GLSxtrlongplformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but all caps.

\glsxtrshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

This command is used in the definition of \glsxtrshort and in some of the predefined abbreviation styles to format the short value of the entry identified by ⟨*entry-label*⟩ with the command ⟨*fmt-cs*⟩, which should take one argument.

Accessibility support is implemented with \glsaccessshort if the markshortwords attribute is true otherwise with \glsaccessfmtshort using \glsxtrgenentrytextfmt for the inner formatting.

This is then encapsulated (including or excluding the ⟨*insert*⟩, according to \ifglsxtrinsertinside) with ⟨*fmt-cs*⟩. If the ⟨*insert*⟩ content needs to be placed outside of ⟨*fmt-cs*⟩, it will be individually encapsulated with the inner formatting.

\Glsxtrshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but sentence case.

\GLSxtrshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but all caps.

\glsxtrshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As \glsxtrshortformat, but for the shortplural field.

\Glsxtrshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but sentence case.

\GLSxtrshortplformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but all caps.

\glsxtrshortformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As \glsxtrshortformat, but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\Glsxtrshortformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but sentence case.

\GLSxtrshortformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but all caps.

\glsxtrshortplformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As \glsxtrshortplformat, but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\Glsxtrshortplformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but sentence case.

\GLSxtrshortplformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}

As above, but all caps.

\glsxtrlongshortformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩}

A shortcut designed for ⟨*long*⟩ (⟨*short*⟩) styles. This is defined as:

```
\glsxtrlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}%
\glsxtrfullsep{⟨entry-label⟩}%
\glsxtrparen{\glsxtrshortformat{⟨entry-label⟩}{}{⟨short-fmt-cs⟩}}
```

Note that the ⟨*insert*⟩ is only placed after the long form.

\Glsxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\glsxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As \glsxtrlongshortformat but uses the plural versions \glsxtrlongplformat and \glsxtrshortplformat.

\Glsxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

A shortcut designed for ⟨*short*⟩ (⟨*long*⟩) styles. This is defined as:

```
\glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨short-fmt-cs⟩}%
\glsxtrfullsep{⟨entry-label⟩}%
\glsxtrparen{\glsxtrlongformat{⟨entry-label⟩}{}{⟨long-fmt-cs⟩}}
```

Note that the ⟨*insert*⟩ is only placed after the short form.

The syntax is the same as for \glsxtrlongshortformat even though \glsxtrlongformat and \glsxtrshortformat are flipped within the definition.

\Glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\glsxtrshortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As `\glsxtrshortlongformat` but uses the plural versions `\glsxtrshortplformat` and `\glsxtrlongplformat`.

\Glsxtrshortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

4.6. Restoring Base Acronym Mechanism

It's possible to revert `\newacronym` back to the definition provided by the base glossaries package. However, if you do this, you will lose all the abbreviation features provided by glossaries−extra.

Where possible, avoid this. If you're simply trying to make the long form appear on first use with `\newacronym`, set the abbreviation style using:

```
\setabbreviationstyle[acronym]{long-short}
```

If you really need to use the original base glossaries package's acronym mechanism, it's better to stick with just glossaries and not use glossaries-extra. However, it may be that you need to use a glossaries-extra feature, such as \printunsrtglossary, but you have a custom acronym style that you can't implement using the glossaries-extra abbreviation mechanism. This is a rare edge case for unusual formats, as it should be possible to implement most common abbreviation formats using the predefined styles.

Unpredictable results will occur if \RestoreAcronyms or \MakeAcronymsAbbreviations are used after abbreviations or acronyms have been defined.

\RestoreAcronyms

Restores \newacronym back to the original base glossaries interface. Note that this doesn't affect \newabbreviation. It also sets the regular attribute for the acronym category to false and sets the acronym style to long-short (which is the default for the base package).

The display style for each glossary identified in the acronym lists is switched to the default acronym display style.

\MakeAcronymsAbbreviations

Counteracts \RestoreAcronyms.

5. Referencing (Using) Entries

Entries can be referenced using the \gls-like and \glstext-like commands, as described in the base glossaries manual. There are some additional commands provided by glossaries−extra:

- abbreviation commands, such as \glsxtrshort (see §4);
- commands for use in captions or section headings, such as \glsfmttext (see §5.3.2);
- commands, such as \glsxtrp, designed for use within fields to help mitigate the problems of nesting (see §5.4);
- commands, such as \mgls, that are designed for referencing multi (or compound) entries (see §7);
- commands, such as \glsaccessname, used to incorporate accessibility support (see §9.2);
- commands, such as \dgls, that are designed for bib2gls's dual entries (see §11.6.7);
- commands, such as \rgls, that depend on the number of entry records (see §11.5).

Additionally, the entry counting commands, such as \cgls, provided by the base glossaries package are modified by glossaries−extra (see §6.1).

The \gls-like commands are designed to produce text at that point in the document (the link text, §5.5), index the entry (to ensure that it appears in the glossary, §5.8) and unset the first use flag (which can alter the link text, §5.10). Additional information can be appended automatically with the post-link hook (§5.5.4). The link text is given by the entry style (see §5.5.5) or by the final argument of \glsdisp.

The \glstext-like commands are designed to produce text at that point in the document (the link text, §5.5) and index the entry (to ensure that it appears in the glossary, §5.8). Additional information can be appended automatically with the post-link hook (§5.5.4). The link text is determined by the calling command. For example, the corresponding field value (possibly encapsulated with \glsxtrregularfont and the inner formatting) for commands like \glstext or the final argument of \glslink.

The \gls-like and \glstext-like commands can all be used with a star (*) or plus (+) modifier. The star modifier automatically implements hyper=false (disables the hyperlink) and the plus modifier automatically implements hyper=true (forces the hyperlink on, if supported).

With glossaries−extra, it's possible to define an additional modifier (⟨*alt-mod*⟩) for your own use with:

```
\GlsXtrSetAltModifier{⟨token⟩}{⟨options⟩}
```

The ⟨*token*⟩ must be a single token, so a multi-byte UTF-8 character will required a native Unicode engine (XeLaTeX or LuaLaTeX). For example, the following:

```
\GlsXtrSetAltModifier{!}{format=glsignore}
```

means that `\gls!{`⟨*label*⟩`}` will be equivalent to :

```
\gls[format=glsignore]{⟨label⟩}
```

It's also possible to redefine the star and plus modifiers:

```
\GlsXtrSetStarModifier{⟨options⟩}
```

This sets the options to use for the star modifier.

```
\GlsXtrSetPlusModifier{⟨options⟩}
```

This sets the options to use for the plus modifier. For example, the following:

```
\GlsXtrSetPlusModifier{noindex}
```

means that the plus modifier will now suppress indexing instead of switching on the hyperlink.

The `\gls`-like and `\glstext`-like commands have a complicated internal structure, which can be viewed as a series of layers. The outermost common layer is:

```
% save settings
% initialise options, see §5.1
\glslinkwrcontent{⟨index & fmt content⟩}
% restore settings
% post-link hook, see §5.5.4
```

The ⟨*index & fmt content*⟩ consists of the indexing (see §5.8) and the (possibly hyperlinked) formatted text, see §5.5. The ⟨*index & fmt content*⟩ code is encapsulated with:

```
\glslinkwrcontent{⟨code⟩}
```

In v1.48, this was added to scope the link text and indexing code, which helped to prevent unwanted spacing caused by the whatsit and also helped to prevent some setting leakage, in the event of nesting (see §5.4), but this caused spacing issues when used in math mode, so from v1.49 this command now simply does its argument. The whatsit is now scoped with `\glsencapwrcontent` instead.

The `\glsxtrp` command, designed for nested use, deals with the problem by suppressing the post-link hook and adding an outer group. For example, `\glsxtrp{short}{html}` behaves like:

```
{\let\glspostlinkhook\relax
 \glsxtrshort[noindex,hyper=false]{html}}
```

Note that the code to suppress the post-link hook has been moved to `\glsxtrpInit`, so it is now possible to allow the post-link hook but it won't be able to lookahead beyond the added outer group.

Depending on the settings (the `wrgloss` option or the `wrgloss` attribute), the indexing may come before the text:

```
\glslinkwrcontent{⟨index⟩⟨fmt content⟩}
```

or after the text:

```
\glslinkwrcontent{⟨fmt content⟩⟨index⟩}
```

or may be suppressed with `noindex`:

```
\glslinkwrcontent{⟨fmt content⟩}
```

The ⟨*fmt content*⟩ part is described in §5.5. The ⟨*index*⟩ part is the actual indexing (see §5.8) but also increments the index count, if applicable. Both the associated whatsit and increment are encapsulated with `\glsencapwrcontent`.

Avoid using `\glstext`, `\glsplural`, `\glsfirst` and `\glsfirstplural` (and their case-changing variants) with entries that have been defined with `\newabbreviation`. Some of the abbreviation styles are too complicated to work with those commands. Instead, use commands like `\glsxtrshort`, `\glsxtrfull` or use `\gls` with the `prereset` or `preunset` options.

The base glossaries package provides a way to adjust the formatting of the link text for the `\gls`-like commands according to the glossary type with `\defglsentryfmt`. The glossaries−extra package changes the default entry formatting (§5.5.5) and provides additional ways of modifying the displayed content (§5.5).

The heading commands (described in §5.3) are designed to prevent indexing or changes to the first use flag if they appear in the table of contents (or list of figures, etc) or if they appear in the page header.

Although the base glossaries package warns against nested link text, the glossaries-extra package provides \glsxtrp which can be used instead of \gls in field values to overcome some of the associated problems. See §5.4 for further details.

If you need to simply access a field value without any formatting, see §5.11. (See §3.5 to set field values.) If you want to encapsulate the value with the appropriate accessibility tag, see §9.2.

Commands such as \glsadd (see §5.8) and \glssee (see §5.9) are designed to only index (to ensure the entry appears in the glossary) without producing any text or changing the first use flag.

The \gls-like, \glstext-like and \glsadd commands all have an initial optional argument that can be used to override the default actions. Some options are only applicable for particular subsets of referencing commands. For example, noindex is pointless for \glsadd since the sole purpose of that command is to index. Whereas types is only available with \glsaddall.

5.1. Options

Some options are provided by the base glossaries package, but there are some additional options provided by glossaries-extra, which are listed in §5.1.2. Below, ⟨*option-list*⟩ indicates the options that are passed in the optional argument of the calling command (such as \gls).

The order that the options are applied is:

1. prereset, preunset and postunset options are initialised by \glsinitreunsets;
2. hyper is initialised by \glsxtrchecknohyperfirst (\glsfirst-like only);
3. wrgloss option is initialised by \glsxtrinitwrgloss (not implemented by \glsadd or \glsxtrfmt);
4. hyperoutside option is initialised by \glsxtrinithyperoutside (not implemented by \glsadd or \glsxtrfmt);
5. initialise noindex=false (not \glsadd);
6. options that have been identified with \GlsXtrSetDefaultGlsOpts, \GlsXtrAppToDefaultGlsOpts or \GlsXtrPreToDefaultGlsOpts (not implemented by \glsadd);
7. (\glsxtrfmt only) options provided in \GlsXtrFmtDefaultOptions;
8. (\gls-like only) the hyperfirst package option, nohyperfirst attribute and nohypernext attributes are checked to determine if the hyper option should be switched off (tests followed by \glslinkcheckfirsthyperhook);

9. \glslinkpresetkeys (not implemented by \glsadd or \glsxtrfmt);
10. (\glsadd only) \glsaddpresetkeys;
11. ⟨*option-list*⟩;
12. \glslinkpostsetkeys (provided by the base glossaries package, not implemented by \glsadd or \glsxtrfmt);
13. (\glsadd only) \glsaddpostsetkeys.

5.1.1. Setting Up Defaults

You can (locally) set your preferred default options for the \gls-like and \glstext-like commands using:

```
\GlsXtrSetDefaultGlsOpts{⟨options⟩}
```

The ⟨*options*⟩ may be any options that you can pass to those commands. These options also apply to \glsxtrfmt but not to \glsadd.

Note that multiple instances of \GlsXtrSetDefaultGlsOpts will override each other.

If you want to add to the existing options, you can use one of the following commands (both may be scoped).

```
\GlsXtrAppToDefaultGlsOpts{⟨options⟩}
```

Appends ⟨*options*⟩ to the list of default options.

```
\GlsXtrPreToDefaultGlsOpts{⟨options⟩}
```

Prepends ⟨*options*⟩ to the list of default options.

For example, to prevent indexing in the front matter and back matter but not in the main matter:

```
\frontmatter
\GlsXtrSetDefaultGlsOpts{noindex}
...
\mainmatter
```

```
\GlsXtrSetDefaultGlsOpts{}
...
\backmatter
\GlsXtrSetDefaultGlsOpts{noindex}
```

Note that `noindex=false` is now set before the options given in `\GlsXtrSetDefaultGlsOpts` to ensure that the setting is correctly initialised, so as from v1.49 you can simply set an empty options list to reset the default. Prior to v1.49, it was necessary to ensure that the `noindex` key was always present in the options list to avoid instability. So for pre v1.49, the line after `\mainmatter` in the above would need to be:

```
\GlsXtrSetDefaultGlsOpts{noindex=false}
```

The default location encap is `glsnumberformat` but can be changed (locally) with:

```
\GlsXtrSetDefaultNumberFormat{⟨encap⟩}
```

This can be overridden by explicitly setting the `format` key.

The default options for `\glsxtrfmt` only are given by:

`\GlsXtrFmtDefaultOptions` *initial:* `noindex`

This command should simply expand to the required list of options. These options are set after any options given in `\GlsXtrSetDefaultGlsOpts` and before ⟨*option-list*⟩.

```
\glslinkpresetkeys
```

This hook is performed after any settings provided in `\GlsXtrSetDefaultGlsOpts` but before ⟨*option-list*⟩. This hook also applies to `\glsxtrfmt` but not to `\glsadd`.

Note that `\glslinkpostsetkeys`, provided by the base glossaries package, is performed after ⟨*option-list*⟩ is processed.

```
\glsaddpresetkeys
```

This hook, which is only used by `\glsadd`, is performed before ⟨*option-list*⟩.

```
\glsaddpostsetkeys
```

This hook, which is only used by `\glsadd`, is performed after ⟨*option-list*⟩.

```
\glsinitreunsets
```

This hook initialises the pre unset/reset options to: prereset=none and preunset=none. It also initialises the postunset setting to perform the post-unset (where applicable) but it will retain the current local/global setting.

This hook will also implement the local repeat unset feature of \GlsXtrUnsetBufferEnableRepeatLocal.

```
\glsxtrchecknohyperfirst{⟨entry-label⟩}
```

This hook is only used by \glsfirst, \glsfirstplural and their case-changing variants. The hook will implement hyper=false if the nohyperfirst attribute is set to true.

```
\glsxtrinitwrgloss
```

This hook initialises the default setting of the wrgloss option. If the wrgloss attribute is set to after then this implements wrgloss=after otherwise it implements wrgloss=before. This setting can subsequently be overridden by \GlsXtrSetDefaultGlsOpts, \glslinkpresetkeys, the ⟨*option-list*⟩ argument or \glslinkpostsetkeys. This hook also applies to \glsxtrfmt but not to \glsadd.

If you prefer to have the default to place the indexing after the link text, you can redefine this hook as follows:

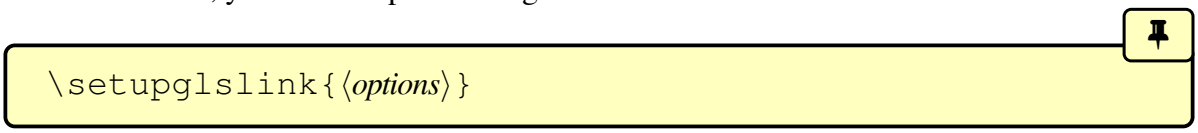
```
\renewcommand*{\glsxtrinitwrgloss}{%
 \glsifattribute{\glslabel}{wrgloss}{before}%
 {%
   \glsxtrinitwrglossbeforetrue
 }%
 {%
   \glsxtrinitwrglossbeforefalse
 }%
}
```

```
\glsxtrinithyperoutside
```

This hook initialises the default setting of the hyperoutside option. If the hyperoutside attribute is set to false then this implements hyperoutside=false otherwise it implements hyperoutside=true. This setting can subsequently be overridden by

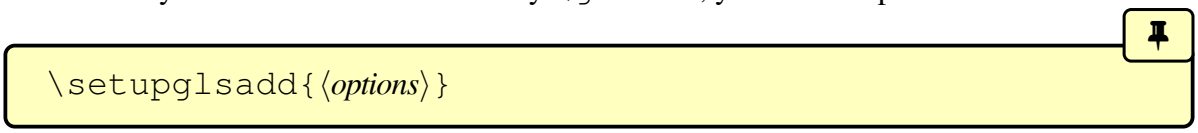
\GlsXtrSetDefaultGlsOpts, \glslinkpresetkeys, the ⟨*option-list*⟩ argument or \glslinkpostsetkeys. This hook also applies to \glsxtrfmt but not to \glsadd.

Within any of the hooks that are used by the \gls-like, \glstext-like or \glsxtrfmt commands, you can set options using:



```
\setupglslink{⟨options⟩}
```

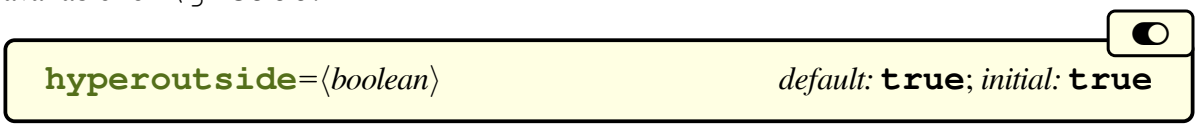
Within any of the hooks that are used by \glsadd, you can set options with:



```
\setupglsadd{⟨options⟩}
```

5.1.2. Additional Options

Options for the \gls-like and \glstext-like commands that are provided by the base glossaries package also apply to new commands like \glsxtrfmt and \glsfmttext. In addition, the options below are provided by glossaries-extra. Note that some options, such as postunset, only apply to the \gls-like commands. Options that relate to the hyperlink, formatting, first use flag or whether/where (noindex/wrgloss) to perform indexing aren't available for \glsadd.

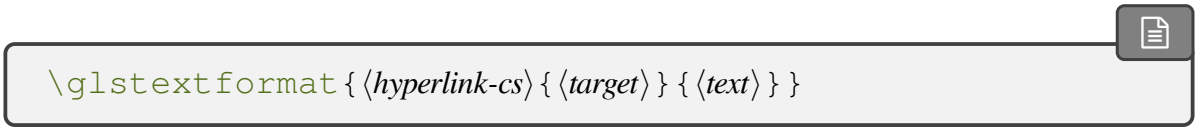


hyperoutside=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

This boolean option determines whether the hyperlink should be inside or outside of \glstextformat (see §5.5.1). If true, the link text is encapsulated as:

```
⟨hyperlink-cs⟩{⟨target⟩}{\glstextformat{⟨text⟩}}
```

otherwise it's encapsulated as:



```
\glstextformat{⟨hyperlink-cs⟩{⟨target⟩}{⟨text⟩}}
```

where ⟨*hyperlink-cs*⟩ is the command that generates the hyperlink (if enabled).

textformat=⟨*csname*⟩

The value of this key should be the name of a control sequence (without the leading backslash). If this option is set, the given control sequence will be used instead of `\glstextformat` to encapsulate the link text. Note that this control sequence should take a single argument (the link text). See §5.5.1 for further details.

This option will override the `textformat` attribute.

innertextformat=⟨*csname*⟩ *initial:* **glsxtrdefaultentrytextfmt**

The value of this key should be the name of a control sequence (without the leading backslash). The command `\glsxtrgenentrytextfmt` (which shouldn't be redefined) is assigned to this control sequence at the start of the `\gls`-like and `\glstext`-like commands. This command is used within the predefined abbreviation styles and within `\glsgenentryfmt` to encapsulate the entry field values.

Custom styles that don't use `\glsxtrgenentrytextfmt` won't support this key. See §5.5.3 for further details.

Some formatting commands require direct access to the actual text or else the content has to be placed inside a box (which inhibits line-breaking). These commands won't work with `textformat` as the text is usually too deeply embedded. This option provides a way of using those problematic commands, however there's still no guarantee that they will work (for example, in the case of custom styles or where the field value itself contains commands).

postunset=⟨*value*⟩ *default:* **global**; *initial:* **global**

This option only applies to the `\gls`-like commands and indicates whether or not to unset the first use flag after the link text. It may take one of three values: `global` (behaves like `local=false`), `local` (behaves like `local=true`) or `none` (doesn't unset the first use flag after the first use). See §5.10.

prereset=⟨*value*⟩ *default:* **local**; *initial:* **none**

This option may take one of three values: `none` (no reset), `local` or `global`. This option (if not `none`) will reset the first use flag before the link text and additionally change `\glsxtrifwasfirstuse` so that it indicates that this was the first use of the entry. See §5.10.

Note that this is different from using \glslocalreset or \glsreset before the \glstext-like commands. Normally \glstext and \glsplural will define \glsxtrifwasfirstuse so that it indicates that this was not the first use of the entry (regardless of whether or not the entry has actually been used).

For example:

```
\glsdefpostlink
 {general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},
  first={sample first use},description={an example}}
\begin{document}
Text field: \glstext{sample}.

First use: \gls{sample}.  Next use: \gls{sample}.

Force reset: \gls[prereset]{sample}.
Used? \ifglsused{sample}{Yes}{No}.

Force reset: \glstext[prereset]{sample}.
Used? \ifglsused{sample}{Yes}{No}.
\end{document}
```

Example 90: Illustrating the prereset option

Text field: sample.
First use: sample first use (an example). Next use: sample.
Force reset: sample first use (an example). Used? Yes.
Force reset: sample (an example). Used? No.

Note that \gls unsets the first use flag (unless postunset=none), so the sample entry is marked as used afterwards, but \glstext doesn't alter the first use flag, after the link text so the sample entry is still marked as unused afterwards.

preunset=⟨*value*⟩ *default:* **local**; *initial:* **none**

This option may take one of three values: none (no unset), local or global. This option (if not none) will unset the first use flag before the link text and additionally change \glsxtrifwasfirstuse so that it indicates that this wasn't the first use of the entry. See §5.10.

The preunset key is always performed after the prereset key.

Note the effect of using a global reset but a local unset in the example below. Both options are performed, but the unset locally overrides the global reset.

```
\glsdefpostlink
 {general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},
  first={sample first use},description={an example}}
\begin{document}
\gls{sample}. Used? \ifglsused{sample}{Yes}{No}.

{\glsfirst[preunset=local,prereset=global]{sample}.
Used? \ifglsused{sample}{Yes}{No}.
}

Used? \ifglsused{sample}{Yes}{No}.

{\gls[preunset=local,prereset=global]{sample}.
Used? \ifglsused{sample}{Yes}{No}.
}

Used? \ifglsused{sample}{Yes}{No}.
\end{document}
```

Example 91: Combining prereset and preunset

sample first use (an example). Used? Yes.
sample first use. Used? Yes.
Used? No.
sample. Used? Yes.
Used? Yes.

Remember that \gls globally unsets the first use flag (unless changed with postunset), which counteracts prereset=global.

noindex=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

This is a boolean option that determines whether or not to suppress the normal indexing. For example, to prevent any locations in the front matter or back matter appearing in the glossary:

```
\frontmatter
\GlsXtrSetDefaultGlsOpts{noindex}
...
\mainmatter
\GlsXtrSetDefaultGlsOpts{noindex=false}
...
\backmatter
\GlsXtrSetDefaultGlsOpts{noindex}
```

Note that if you are using auto-indexing (see §12), `noindex=false` will also suppress the auto-indexing.

If you are using `bib2gls`, you may want to consider instead using `format=glsignore` to create an ignored location that ensures the entry is selected without adding a location to the location list. (Don't use this method for the other indexing methods as you'll end up with invisible locations with spurious commas in your location lists.)

wrgloss=⟨*position*⟩ *initial:* **before**

This option may take one of two values, `before` or `after`, which indicate whether the indexing should occur before or after the link text. The indexing creates a whatsit that can interfere with spacing or cause other problems. The other thing to consider is where the link text is long, such as a phrase or full form of an abbreviation, that may be split by a page break. You will need to decide if you want the indexing before the link text, so that the location is at the end of the page where the text starts, or if you want the indexing after the link text, so that the location is at the start of the next page where the text ends.

This option corresponds to a conditional:

```
\ifglsxtrinitwrglossbefore ⟨true⟩\else ⟨false⟩\fi
                                        initial: \iftrue
```

The hook `\glsxtrinitwrgloss` sets this conditional according to whether or not the `wrgloss` attribute has been set to `after`:

```
\newcommand*{\glsxtrinitwrgloss}{%
 \glsifattribute{\glslabel}{wrgloss}{after}%
 {\glsxtrinitwrglossbeforefalse}%
 {\glsxtrinitwrglossbeforetrue}%
}
```

thevalue=⟨*location*⟩

Sets the entry location to the given value instead of obtaining it from the location counter. If you are using hyperref you may also need to set the location's hypertarget with `theHvalue`.

This option is primarily intended for use with `bib2gls` to supply locations that don't have an associated counter within the document, such as an external location. If you want to automatically add locations from a supplemental document to an entry's location list, you can use the `supplemental-locations` resource option. See the `bib2gls` user manual for further details.

For example, to index a location in a supplementary document:

```
\glsadd[thevalue={Suppl.\ 2.45}]{sample}
```

This will add "Suppl. 2.45" to the location list for the "sample" entry.

Note that the value must conform to the indexing application's location syntax. For `makeindex`, this is limited to `Roman`, `roman`, `arabic`, `alph` and `Alph`. With `xindy`, the location syntax must be defined in the `xindy` module (standard location syntaxes are supplied by default). There's no restriction on the location syntax for `bib2gls`, although if it can't deduce a numerical value it won't be able to form a range.

If you want a hyperlink to an external file, you can use:

\glsxtrsupphypernumber{⟨*location*⟩}

as the formatting command for the location encap. For example:

```
\glsadd[thevalue=S.2,format=glsxtrsupphypernumber]
{sample}
```

The path to the external file needs to be set in the `externallocation` category attribute.

The hyperlink for the supplementary location may or *may not* take you to the relevant place in the external PDF file *depending on your PDF viewer*. Some may not support external links, and some may take you to the first page or last visited page.

For example, if both the file `sample-suppl-hyp.pdf` and the file `sample-suppl-main-hyp.pdf` are in the same directory, then viewing the file `sample-suppl-main-hyp.pdf` in Evince will take you to the correct location in the linked document (when you click on the S.2 external link), but Okular will take you to the top of the first page of the linked document.

This method can only be used where there is one external source for the designated category (identified by the `externallocation` attribute). For multiple sources, you need to use `bib2gls` v1.7+, which is the better method in general as it can automatically fetch the relevant locations from the `aux` files of the designated external documents without the need to explicitly use `\glsadd`.

theHvalue=⟨*the-H-value*⟩

Sets the hypertarget corresponding to the location, which will be used if the `format` supports hyperlinks. This is analogous to hyperref's `\the`⟨*counter-name*⟩ that provides the hypertarget for a reference to `\the`⟨*counter-name*⟩.

This option is primarily intended for use with the `thevalue` option.

Unless you are using `record=nameref`, you must ensure that it's possible to form ⟨*the-H-value*⟩ from ⟨*h-prefix*⟩⟨*thevalue*⟩ for some ⟨*h-prefix*⟩ (where ⟨*thevalue*⟩ is given by `thevalue` or the value of the location counter). This restriction is due to the limitations imposed by `makeindex` and `xindy`.

prefix=⟨*link-prefix*⟩

This option locally redefines `\glolinkprefix` to ⟨*link-prefix*⟩. If you are using `\printunsrtglossary` to redisplay a list (possibly in a different order) then you will need some way of changing the entry targets to avoid duplicate hyperlink targets. One way of achieving this is to redefine `\glolinkprefix` for the subsequent lists. You will then need to use the `prefix` option in commands like `\gls` to ensure that the hyperlink for the link text points to the desired list.

This option is intended for use with the "unsrt" family of commands and `\glsxtrcopytoglossary` (which is used by `bib2gls`). The other indexing methods don't support repeated lists.

5.2. Case Changing

Case-changing commands, such as \Gls and \GLS, perform the conversion using commands provided by mfirstuc. The underlying commands provided by mfirstuc were redesigned in v2.08 to use the newer, better case-changing commands available with the LaTeX3 kernel. The base glossaries package v4.50 and glossaries−extra v1.49 were developed concurrently with mfirstuc v2.08 to take advantage of the new features. Version 1.49 of glossaries−extra was also developed concurrently with bib2gls v3.0 which, in turn, was developed alongside version 0.9.2.7b of the TeX parser library.

It's not possible to upload all these new versions at the same time, so it will be necessary to stagger their deployment. The new case-changing features will work best when all these new versions are installed. In the interim, a reduced feature set will be used.

5.2.1. Sentence Case Commands

Both the base glossaries package and the glossaries−extra package provide sentence case commands, which convert the first letter to uppercase. These are provided for situations where an entry is referenced at the start of a sentence. Sentence-casing is also implemented when the attributes glossname or glossdesc are set to firstuc.

The case conversion is performed using:

```
\glssentencecase{⟨text⟩}
```

The default definition uses \makefirstuc, which is provided by the mfirstuc package. This was originally part of the base glossaries package, but was split into a separately distributed package in 2015. Back then, there was no expandable sentence-case command. There was also a problem with referencing entries where link text was encapsulated with a text-block command (which occurs, in particular, with acronym and abbreviation styles). The first letter of the text-block command's argument needed to be obtained, which resulted in some trickery that proved problematic with UTF-8.

The LaTeX3 kernel now provides a suitable expandable command that works with UTF-8, and mfirstuc v2.08+ provides \MFUsentencecase that directly interfaces with it. If an older version of mfirstuc is installed, glossaries v4.50+ and glossaries−extra v1.49+ will provide \MFUsentencecase. You can use this in expandable contents. For example:

```
\section{\MFUsentencecase{\glsentrytext{label}}}
```

However, in the above example, it's simpler to do:

```
\section{\Glsfmttext{label}}
```

If hyperref has been loaded, \Glsfmttext{label} will now expand to:

```
\MFUsentencecase{\glsentrytext{label}}
```

in the PDF bookmark.

Internally, \makefirstuc now uses \MFUsentencecase to perform the case conversion, but it still parses its argument to determine if it starts with ⟨*cs*⟩{⟨*text*⟩}. This means that with mfirstuc v2.08+, you now don't have to worry about UTF-8 characters occurring at the start of the text.

For example, with mfirstuc v2.07 you would need to do something like:

```
\newglossaryentry{elite}{name={{é}lite},
 description={...}}
```

in order for \Gls{elite} to work. Whereas with mfirstuc v2.08, you can now simply do:

```
\newglossaryentry{elite}{name={élite},
 description={...}}
```

(As from glossaries v4.47, it should be possible to use UTF-8 characters in the label as well.)

Whilst you can redefine \glssentencecase to use \MFUsentencecase directly (without using \makefirstuc as an intermediary), this may result in content being expanded that wouldn't have been expanded previously. In particularly, if ⟨*cs*⟩ isn't robust and expands to content that includes labels then the case-change can fail. You also won't be able to take advantage of the blockers and mappings that are only recognised as such by \makefirstuc. If you use \MFUsentencecase instead, blockers and mappings will be treated as exclusions, which are likely to result in unwanted side-effects.

Both \makefirstuc and \MFUsentencecase recognise exclusions. These are text-block commands which take a single mandatory argument that needs to be skipped. For example, in the following \glsadd{example} needs to be skipped:

```
\MFUsentencecase{\glsadd{example}some text}
```

Exclusions are identified with \MFUexcl. If you have an older version of mfirstuc, this won't be defined, so glossaries v4.50+ and glossaries-extra v1.49+ provide:

```
\glsmfuexcl{⟨cs⟩}
```

This will use \MFUexcl with mfirstuc v2.08+. With older versions, a definition will be provided that works with \MFUsentencecase, but exclusions won't be recognised by \make-

firstuc.

As from glossaries v4.50, \glsadd will be identified as an exclusion (via \glsmfuexcl), but the optional argument will cause a problem if present. See the mfirstuc v2.08+ manual for a workaround. Note that commands such as \glsaddall and \glsaddeach aren't identified as exclusions as they aren't expected to occur in text that may require a case-change.

With glossary entry references, there are commands that take a label as the argument, which shouldn't have any case-changed applied, but also shouldn't be skipped. For example:

```
\makefirstuc{\GLS{example} something}
```

In this situation, there shouldn't be any case-change as \GLS already implements a case-change. This type of command is referred to as a blocker in the mfirstuc manual, as it indicates a command that should prevent any case-change if it's encountered at the start of the text. Blockers are identified with \MFUblocker. If you have an older version of mfirstuc, this won't be defined, so glossaries v4.50+ and glossaries−extra v1.49+ provide:

```
\glsmfublocker{⟨cs⟩}
```

This will use \MFUblocker with mfirstuc v2.08+. With older versions, it will simply use \glsmfuexcl which will instead identify the command as an exclusion and won't be recognised by \makefirstuc. See the mfirstuc v2.08+ manual for further information about blockers.

As from glossaries v4.50+, commands like \GLS will be identified as blockers using \glsmfublocker, and glossaries−extra now identifies similar commands, such as \rGLS as blockers.

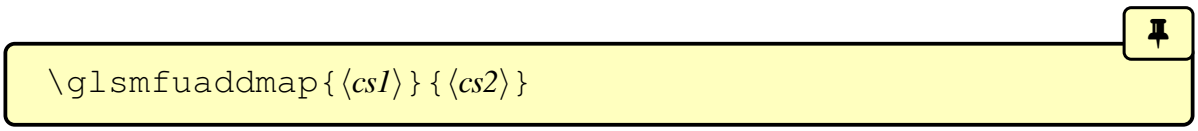
Finally, there are mappings. These are commands that should be substituted with another command, which is expected to perform the case-change. For example:

```
\makefirstuc{\gls{example} something}
```

This shouldn't skip or block \gls but instead should convert the text to:

```
\Gls{example} something
```

This is implemented by adding a mapping from \gls to \Gls. Mappings are added using \MFUaddmap. If you have an older version of mfirstuc, this won't be defined, so glossaries v4.50+ and glossaries−extra v1.49+ provide:



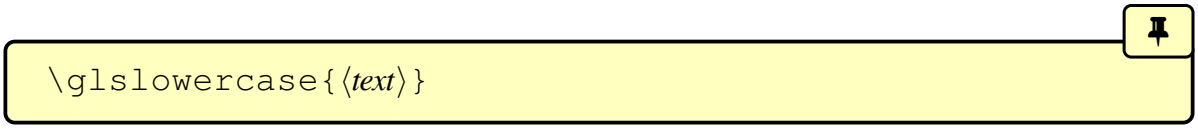
\glsmfuaddmap{⟨cs1⟩}{⟨cs2⟩}

This will use \MFUaddmap with mfirstuc v2.08+. With older versions, it will simply use \glsmfuexcl which will instead identify the command as an exclusion and won't be recognised by \makefirstuc. See the mfirstuc v2.08+ manual for further information about mappings.

As from glossaries v4.50+, commands like \gls will be mapped to the appropriate sentence case command using \glsmfuaddmap, and glossaries-extra now identifies similar mappings, such as \rgls mapped to \rGls.

In order to integrate the full set of features provided by mfirstuc v2.08+, you will need both glossaries v4.50+ and glossaries-extra v1.49+.

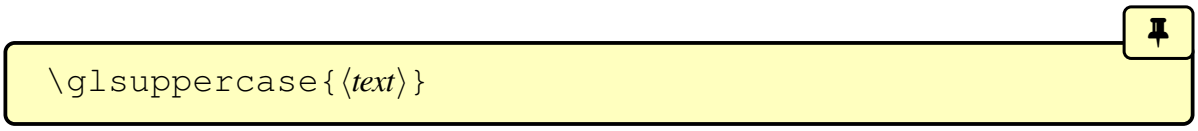
5.2.2. Lower Case



\glslowercase{⟨text⟩}

This is defined by glossaries v4.50+ to use the LaTeX3 command to convert to lowercase. If an older version of glossaries is present, then this command will be provided by glossaries-extra but it will be defined to use \MakeTextLowercase instead. This command is primarily provided for use with small caps styles to convert an abbreviation to lowercase, but isn't actually used anywhere by default.

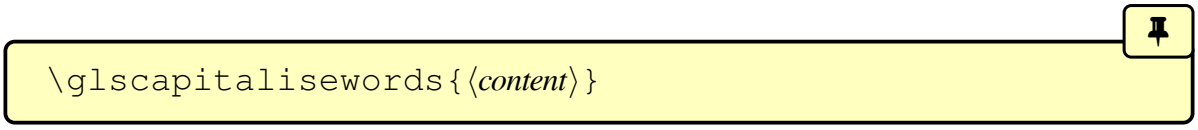
5.2.3. Upper Case



\glsuppercase{⟨text⟩}

This is defined by glossaries v4.50+ to use the LaTeX3 command to convert to uppercase (all caps). If an older version of glossaries is present, then this command will be provided by glossaries-extra but it will be defined to just use \mfirstucMakeUppercase, which is provided by mfirstuc. This command is used by all caps commands such as \GLSxtrusefield.

5.2.4. Title Case



\glscapitalisewords{⟨content⟩}

This is defined by glossaries v4.48 to use \capitalisewords to convert to title case. If you experience any errors with title case commands, such as \glsentrytitlecase, or

attributes such as glossdesc then try redefining this command to use \capitalisefmtwords* instead. See the mfirstuc manual for further details.

```
\glspdfsentencecase{⟨text⟩}
```

This is provided for use in the PDF part of \texorpdfstring, and is used by commands such as \Glsfmtlong. It ensures that the ⟨text⟩ argument is expanded (with \exp_args:Ne) before applying \MFUsentencecase.

5.3. Entries in Sectioning Titles, Headers, Captions and Contents

The glossaries user manual cautions against using commands like \gls in chapter or section titles. The principle problems are:

- if you have a table of contents, the first use flag will be unset in the contents rather than later in the document;
- if you have the location lists displayed in the glossary, unwanted locations will be added to it corresponding to the table of contents (if present) and every page that contains the entry in the page header (if the page style in use adds the chapter or section title to the header);
- if the page style in use adds the chapter or section title to the header and attempts to convert it to uppercase, the entry label (in the argument of \gls etc) will be converted to uppercase and the entry won't be recognised;
- if you use hyperref, commands like \gls can't be expanded to a simple string and only the label will appear in the PDF bookmark (with a warning from hyperref);
- if you use hyperref, you will end up with nested hyperlinks in the table of contents.

Similar problems can also occur with captions (except for the page header and bookmark issues).

The glossaries-extra package tries to resolve the header problem by modifying \markright, \markboth and \@starttoc. If this causes unwanted side-effects, you can restore their former definitions using:

```
\glsxtrRevertMarks
```

This will revert \markright, \markboth and \@starttoc back to the definitions in effect when glossaries-extra was loaded. Alternatively, you can use:

```
\glsxtrRevertTocMarks
```

This will only revert \@starttoc.

If you use \glsxtrRevertMarks or \glsxtrRevertTocMarks, you will need to employ the simplistic approach, described in §5.3.1, which is the method recommended by the glossaries user manual. Otherwise, you can use the commands described in §5.3.2, which provide a better solution.

5.3.1. Simplistic Approach

To get around all these problems, the glossaries user manual recommends using the expandable non-hyperlink commands, such as \glsentrytext (for regular entries) or \glsentryshort (for abbreviations). This is the simplest solution, but doesn't allow for special formatting that's applied to the entry through commands like \glstext or \glsxtrshort.

Example 92 uses this approach:

92

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\glsdefpostlink
 {general}{\glsxtrpostlinkAddDescOnFirstUse}

\newglossaryentry{sample}{name={sample},
 description={an example}}
\newglossaryentry{alpha}{
 name={\ensuremath{\alpha}},
 description={alpha}}

\begin{document}
\tableofcontents
\section{\texorpdfstring{\Glsentrytext{sample}
 and \glsentrytext{alpha}}{Sample and alpha}}

First use: \gls{sample} and \gls{alpha}.

Next use: \gls{sample} and \gls{alpha}.

\printunsrtglossary
\end{document}
```

Example 92: References in section headings (simplistic approach)

Contents

1 Sample and α

First use: sample (an example) and α (alpha).
Next use: sample and α.

Glossary

sample an example

α alpha

This solves some problems: it avoids nested links in the table of contents, the first use flag isn't prematurely unset and the PDF bookmarks has a reasonable substitution, but it still isn't a complete solution as the above document will fail if the page style is changed to `headings` and a page break is inserted before the section (after `\tableofcontents`), which will lead to the error:

```
Glossary entry `SAMPLE' has not been defined.
```

This is because the case-change applied to the header converts the label "sample" to "SAMPLE", which doesn't correspond to a defined entry. (This can now be avoided with mfirstuc v2.08+.)

If the case conversion is applied by, then the case-change can be prevented by encapsulating the label with `\NoCaseChange`, but this ends up quite complicated. This is actually what the commands describe in §5.3.2 do when they are in a heading. This allows for older versions of mfirstuc that don't recognise exclusions. See §5.2 for further details.

The `\NoCaseChange` command was originally provided by the textcase package to prevent `\MakeTextUppercase` from applying a case-change. The functionality of the textcase package has now been absorbed into the LaTeX kernel, which means that as from 2022, textcase is deprecated and `\NoCaseChange` is defined by the kernel.

5.3.2. New Commands Designed for Chapter/Section Headings or Captions

This section is irrelevant if you use \glsxtrRevertMarks to restore the definitions of \markright, \markboth and \@starttoc. If you use \glsxtrRevertTocMarks, then this section is only applicable to \markright and \markboth.

The commands listed here are provided for use within captions or section headings. They are designed to overcome some of the problems illustrated in the previous section. Note that they only have a single argument, the entry label. There are no optional arguments. Below, "header" refers to page header text added with \markright or \markboth, and "contents" refers to the table of contents or any other "list of" that uses \@starttoc, such as the list of figures.

Each command \glsfmt⟨*field*⟩ (such as \glsfmttext or \glsfmtshort) behaves like an analogous \gls⟨*field*⟩ or \glsxtr⟨*field*⟩ command (such as \glstext or \glsxtrshort) but with the options noindex and hyper=false and no insert. When they occur within a header, they are protected from having any case-change applied (which will interfere with the entry label). Since this means they won't appear in all caps in the header, the headuc attribute may be set to use the all caps \GLS⟨*field*⟩ or \GLSxtr⟨*field*⟩ instead (such as \GLStext or \GLSxtrshort).

There is currently only support for the name, text, plural, first, firstplural, short, shortplural, long, and longplural fields, and also limited support for the full form of abbreviations. For other fields, you will need to follow the recommendation of the glossaries manual (as discussed above in §5.3.1).

Example 93 is a modification of the previous example:

93

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\glsdefpostlink
 {general}{\glsxtrpostlinkAddDescOnFirstUse}

\newglossaryentry{sample}{name={sample},
 description={an example}}
\newglossaryentry{alpha}{
 name={\ensuremath{\alpha}},
 description={alpha}}

\begin{document}
\tableofcontents
\section{\Glsfmttext{sample} and \glsfmttext{alpha}}
```

```
First use: \gls{sample} and \gls{alpha}.

Next use: \gls{sample} and \gls{alpha}.

\printunsrtglossary
\end{document}
```

Example 93: References in section headings using \glsfmttext

Contents

1 Sample and α

First use: sample (an example) and α (alpha).
Next use: sample and α.

Glossary

sample an example

α alpha

Note that this still results in "Token not allowed in a PDF string" warnings from hyperref. This is due to the maths shift and \alpha, and is something that would also occur if the section title explicitly contained α. If this is likely to happen, the issue can be solved by placing \texorpdfstring within the field value. For example:

```
\glsnoexpandfields
\newglossaryentry{alpha}{description={alpha},
 name={\texorpdfstring{\ensuremath{\alpha}}{alpha}}}
```

Note the need to prevent field expansion with \glsnoexpandfields, otherwise \texorpdfstring will be prematurely expanded while the entry is being defined.

The options noindex and hyper=false are hard-coded when the commands listed below, such as \glsfmtshort, occur in the header or contents, but within the actual section title or caption in the document text, those options are obtained from:

\glsxtrtitleopts

This simply expands to the option list. For example, you may actually want a hyperlink and indexing to occur in the document body, in which case redefine \glsxtrtitleopts to do nothing. Example 94 demonstrates this:

94

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\pagestyle{headings}
\glssetcategoryattribute{general}{headuc}{true}
\glsdefpostlink
 {general}{\glsxtrpostlinkAddDescOnFirstUse}

\newglossaryentry{sample}{name={sample},
 description={an example}}
\renewcommand{\glsxtrtitleopts}{}

\begin{document}
\section{\Glsfmttext{sample}}
First use: \gls{sample}.
Next use: \gls{sample}.
\printunsrtglossary
\end{document}
```

Example 94: Reference with hyperlink in section headings

1 SAMPLE 1

1 Sample

First use: sample (an example). Next use: sample.

Glossary

sample an example

\glsfmtshort{⟨entry-label⟩}

This normally behaves like \glsxtrshort but expands to just \glsentryshort in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtshort{⟨entry-label⟩}

This normally behaves like \Glsxtrshort but expands to:

\MFUsentencecase{\glsentryshort{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtshort{⟨entry-label⟩}

This normally behaves like \GLSxtrshort but expands to just \glsentryshort in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \glsfmtshort but inserts the prefix field and separator, if the prefix value is set and non-empty. Provided for use with glossaries−prefix.

\Pglsfmtshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsfmtshort but sentence case. Note the initial "P" in the command name, which matches \Pgls (similarly for the other prefix sentence case commands).

\PGLSfmtshort[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \pglsfmtshort but all caps.

\glsfmtshortpl{⟨entry-label⟩}

This normally behaves like \glsxtrshortpl but expands to just \glsentryshortpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtshortpl{⟨entry-label⟩}

This normally behaves like \Glsxtrshortpl but expands to:

\MFUsentencecase{\glsentryshortpl{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtshortpl{⟨entry-label⟩}

This normally behaves like \GLSxtrshortpl but expands to just \glsentryshortpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \glsfmtshortpl but inserts the prefixplural field and separator, if the prefixplural value is set and non-empty. Provided for use with glossaries-prefix.

\Pglsfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtshortpl but sentence case.

\PGLSfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtshortpl but all caps.

\glsfmtlong{⟨entry-label⟩}

This normally behaves like \glsxtrlong but expands to just \glsentrylong in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtlong{⟨entry-label⟩}

This normally behaves like \Glsxtrlong but expands to:

\MFUsentencecase{\glsentrylong{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtlong{⟨entry-label⟩}

This normally behaves like \GLSxtrlong but expands to just \glsentrylong in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \glsfmtlong but inserts the prefixfirst field and separator, if the prefixfirst value is set and non-empty. Provided for use with glossaries-prefix.

\Pglsfmtlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtlong but sentence case.

\PGLSfmtlong[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtlong but all caps.

\glsfmtlongpl{⟨entry-label⟩}

This normally behaves like \glsxtrlongpl but expands to just \glsentrylongpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtlongpl{⟨entry-label⟩}

This normally behaves like \Glsxtrlongpl but expands to:

\MFUsentencecase{\glsentrylongpl{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtlongpl{⟨entry-label⟩}

This normally behaves like \GLSxtrlongpl but expands to just \glsentrylongpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \glsfmtlongpl but inserts the prefixfirstplural field and separator, if the prefixfirstplural value is set and non-empty. Provided for use with glossaries-prefix.

\Pglsfmtlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtlongpl but sentence case.

\PGLSfmtlongpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

As \pglsfmtlongpl but all caps.

The full form is slightly different as it doesn't correspond to an individual field but instead is formed from a combination of the short and long fields (the order depending on the abbreviation style). Since it's too complicated to simply expand to the appropriate style, a simple expandable command is provided for the PDF bookmarks:

\glspdffmtfull{⟨entry-label⟩}

This just expands to the long form followed by the short form in parentheses:

```
\newcommand*{\glspdffmtfull}[1]{\glsentrylong{#1}
  (\glsentryshort{#1})}
```

You will need to redefine this if you require the short form first. There is an analogous command for the plural:

\glspdffmtfullpl{⟨entry-label⟩}

This has a similar definition to \glspdffmtfull but uses \glsentrylongpl and \glsentryshortpl.

\glsfmtfull{⟨entry-label⟩}

This normally behaves like \glsxtrfull but expands to just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtfull{⟨entry-label⟩}

This normally behaves like \Glsxtrfull but expands to:

```
\MFUsentencecase{\glspdffmtfull{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtfull{⟨entry-label⟩}
```

This normally behaves like \GLSxtrfull but expands to just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmtfullpl{⟨entry-label⟩}
```

This normally behaves like \glsxtrfullpl but expands to just \glspdffmtfullpl in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmtfullpl{⟨entry-label⟩}
```

This normally behaves like \Glsxtrfullpl but expands to:

```
\MFUsentencecase{\glspdffmtfullpl{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtfullpl{⟨entry-label⟩}
```

This normally behaves like \GLSxtrfull but expands to just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmtname{⟨entry-label⟩}
```

This normally behaves like \glsname but expands to just \glsentryname in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmtname{⟨entry-label⟩}
```

This normally behaves like \Glsname but expands to:

```
\MFUsentencecase{\glsentryname{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtname{⟨entry-label⟩}
```

This normally behaves like \GLSname but expands to just \glsentryname in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmttext{⟨entry-label⟩}
```

This normally behaves like \glstext but expands to just \glsentrytext in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmttext{⟨entry-label⟩}
```

This normally behaves like \Glstext but expands to:

```
\MFUsentencecase{\glsentrytext{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmttext{⟨entry-label⟩}
```

This normally behaves like \GLStext but expands to just \glsentrytext in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmtplural{⟨entry-label⟩}
```

This normally behaves like \glsplural but expands to just \glsentryplural in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmtplural{⟨entry-label⟩}
```

This normally behaves like \Glsplural but expands to:

```
\MFUsentencecase{\glsentryplural{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtplural{⟨entry-label⟩}
```

This normally behaves like \GLSplural but expands to just \glsentryplural in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmtfirst{⟨entry-label⟩}
```

This normally behaves like `\glsfirst` but expands to just `\glsentryfirst` in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmtfirst{⟨entry-label⟩}
```

This normally behaves like `\Glsfirst` but expands to:

```
\MFUsentencecase{\glsentryfirst{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtfirst{⟨entry-label⟩}
```

This normally behaves like `\GLSfirst` but expands to just `\glsentryfirst` in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\glsfmtfirstpl{⟨entry-label⟩}
```

This normally behaves like `\glsfirstplural` but expands to just `\glsentryfirstplural` in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\Glsfmtfirstpl{⟨entry-label⟩}
```

This normally behaves like `\Glsfirstplural` but expands to:

```
\MFUsentencecase{\glsentryfirstplural{⟨entry-label⟩}}
```

in PDF bookmarks and is adjusted when appearing in the header or contents.

```
\GLSfmtfirstpl{⟨entry-label⟩}
```

This normally behaves like `\GLSfirstplural` but expands to just `\glsentryfirstplural` in PDF bookmarks and is adjusted when appearing in the header or contents.

5.3.3. Advanced Commands

This section is intended for advanced users and package developers.

The commands described here are irrelevant if you use `\glsxtrRevertMarks` to restore the definitions of `\markright`, `\markboth` and `\@starttoc`. If you use `\glsxtrRevertTocMarks`, then this section is only applicable to `\markright` and `\markboth`.

If you need to know whether or not some code is inside a header or contents list, you can use:

```
\glsxtrifinmark{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if the command occurs within `\markright`, `\markboth` or `\@starttoc` otherwise does ⟨*false*⟩.

If you need to know whether or not some code is inside a contents list (but not the header), you can use:

```
\glsxtrifintoc{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if the command occurs within `\@starttoc` otherwise it does ⟨*false*⟩. (The modified definition of `\@starttoc` sets `\glsxtrifintoc` to `\@firstoftwo` at the start and to `\@secondoftwo` at the end.)

If you need to know whether or not some code is in the PDF bookmarks or heading, you can use:

```
\glsxtrtitleorpdforheading{⟨title⟩}{⟨PDF bookmarks⟩}{⟨heading⟩}
```

This does the applicable argument depending on whether the command occurs within a title/caption or PDF bookmark or heading.

If this command occurs within the `toc` file, it will do its ⟨*heading*⟩ argument but if `\glsxtrtitleorpdforheading` expands while it's being written to the `toc` file, then it will expand to ⟨*title*⟩.

Example 95 illustrates this, but you will need to view the document in a PDF viewer that shows the bookmarks to also see the PDF part:

95

```
\documentclass{report}
\usepackage[T1]{fontenc}
\usepackage{lipsum}
\usepackage{hyperref}
\usepackage{glossaries-extra}
```

```
\pagestyle{headings}
\begin{document}
\tableofcontents
\chapter{\glsxtrtitleorpdforheading{Title1}{PDF1}
{Heading1}
\glsxtrifinmark{in mark}{not in mark}}
\lipsum[1-5]
\chapter{\protect\glsxtrtitleorpdforheading{Title2}
{PDF2}{Heading2}
\protect\glsxtrifinmark{in mark}{not in mark}}
\lipsum
\end{document}
```

In the first case, `\glsxtrtitleorpdforheading` expands as it's being written to the `toc` file, so it expands to "Title". In the second case, `\glsxtrtitleorpdforheading` is protected so that command is written to the `toc` file. On the next LaTeX, when the table of contents is displayed, this command will expand to "Heading", because it's in the `toc` file. Similarly, in the first case, `\glsxtrifinmark` will expand to "not in mark" as it's written to the `toc` file, but in the second case it's expansion is prevented, so it will expand to "in mark" in the table of contents.

Example 95: Chapter Title or PDF Bookmarks or Heading

If gettitlestring has been loaded (used by nameref to provide `\nameref`) then adjustments for both `\glsxtrtitleorpdforheading` and `\glsxtrifinmark` will be added to `\GetTitleStringDisableCommands`, but bear in mind that you will need to use the following for it to have an effect:

```
\GetTitleStringSetup{expand}
```

The commands described in §5.3.2, such as `\glsfmtshort`, are essentially defined as:

```
\texorpdfstring
 {\glsxtrtitle⟨field⟩{⟨entry-label⟩}}% title
 {\glsentry⟨field⟩{⟨entry-label⟩}}% bookmark
```

If `\texorpdfstring` isn't defined, then the definition is:

```
\glsxtrtitle⟨field⟩{⟨entry-label⟩}
```

For example, `\glsfmtshort` is defined as (with hyperref):

```
\newcommand*{\glsfmtshort}[1]{%
 \texorpdfstring
  {\glsxtrtitleshort{#1}}% TeX
  {\glsentryshort{#1}}% PDF
}
```

This ensures that `\glsfmtshort` expands to just `\glsentryshort` within the PDF bookmarks. Provided the field value doesn't contain any problematic commands, this allows the actual value to be added to the bookmarks.

Some of the case-changing commands, such as `\makefirstuc`, can't expand and therefore aren't appropriate for the bookmarks (which need to be a simple text string without any formatting). However, with newer versions of the LaTeX kernel, there are now expandable commands available. Version 2.08 of mfirstuc takes advantage of these changes and now provides the expandable `\MFUsentencecase`.

This means that the sentence case and all caps commands can now also adjust the field value for the bookmark, whereas previously they didn't. For example, `\Glsfmtshort` is now defined as:

```
\newcommand*{\Glsfmtshort}[1]{%
 \texorpdfstring
   {\Glsxtrtitleshort{#1}}% TeX
   {\MFUsentencecase{\glsentryshort{#1}}}% PDF
}
```

The `\glsxtrtitle⟨field⟩` set of commands all default to the corresponding `\glstext`-like command with the options given by `\glsxtrtitleopts` and an empty insert final argument. These title commands are redefined by `\glsxtrmarkhook` to the corresponding `\glsxtrhead⟨field⟩` command. These "head" commands use `\NoCaseChange` to prevent interference from page headers that convert to all caps (which can inappropriately convert the entry label to all caps). Instead, the `headuc` attribute needs to be set to `true` to use the appropriate all caps command. A shortcut command is provided to test for this attribute:

\glsxtrifheaduc{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}

This is defined as:

```
\newcommand*{\glsxtrifheaduc}[3]{%
 \glsxtrifintoc
  {#3}% in TOC
  {\glsifattribute{#1}{headuc}{true}{#2}{#3}}%
}
```

Since the header commands also end up in the contents, where the all caps conversion should not be applied, the definition includes \glsxtrifintoc to skip the check in the contents.

\glsxtrtitleshort{⟨entry-label⟩}

The normal behaviour of \glsfmtshort. This is redefined by \glsxtrmarkhook to \glsxtrheadshort. The default is:

```
\glsxtrshort[noindex,hyper=false]{⟨entry-label⟩}[]
```

(This is performed indirectly via an internal command that ensures that \glsxtrtitle-opts is expanded before being passed in the optional argument.)

\glsxtrheadshort{⟨entry-label⟩}

Used to display the short form in the page header. This is defined as:

```
\newcommand*{\glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

The sentence case commands also check the headuc attribute.

\Glsxtrtitleshort{⟨entry-label⟩}

The normal behaviour of \Glsfmtshort. This is redefined by \glsxtrmarkhook to \Glsxtrheadshort. The default is:

```
\Glsxtrshort[noindex,hyper=false]{⟨entry-label⟩}[]
```

(Again, this is performed indirectly via an internal command that ensures that \glsxtrtitleopts is expanded before being passed in the optional argument.)

\Glsxtrheadshort{⟨entry-label⟩}

Used to display the sentence case short form in the page header. This is defined as:

```
\newcommand*{\Glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

\GLSxtrtitleshort{⟨entry-label⟩}

The normal behaviour of \GLSfmtshort. This is redefined by \glsxtrmarkhook to \GLSxtrheadshort. The default uses \GLSxtrshort in a similar way to \glsxtrtitleshort and \Glsxtrtitleshort.

\GLSxtrheadshort{⟨entry-label⟩}

Used to display the all caps short form in the page header. In this case, there's no need to check to the headuc attribute, but the label needs to be protected from any potential case-change:

```
\newcommand*{\GLSxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \GLSxtrshort[noindex,hyper=false]{#1}[]%
 }%
}
```

All the similar commands listed below are defined in an analogous way, except for the glossaries−prefix commands, where only the sentence case title version is provided. This is because commands like `\Pglsfmtshort` have to determine whether or not to use `\glsfmtshort` or `\Glsfmtshort` depending on whether or not the prefix has been set. Whereas commands like `\pglsfmtshort` simply need to insert the prefix and separator if set and then use the corresponding `\glsfmtshort`.

`\Pglsxtrtitleshort{`⟨*entry-label*⟩`}`

The normal behaviour of `\Pglsfmtshort`.

`\Pglsxtrtitleshortpl{`⟨*entry-label*⟩`}`

The normal behaviour of `\Pglsfmtshortpl`.

`\Pglsxtrtitlelong{`⟨*entry-label*⟩`}`

The normal behaviour of `\Pglsfmtlong`.

`\Pglsxtrtitlelongpl{`⟨*entry-label*⟩`}`

The normal behaviour of `\Pglsfmtlongpl`.

`\glsxtrtitleshortpl{`⟨*entry-label*⟩`}`

The title plural short form. (Normal behaviour of `\glsfmtshortpl`.)

`\glsxtrheadshortpl{`⟨*entry-label*⟩`}`

The header plural short form. (The behaviour of `\glsfmtshortpl` when it occurs in a header.)

`\Glsxtrtitleshortpl{`⟨*entry-label*⟩`}`

The title plural sentence case short form. (Normal behaviour of \Glsfmtshortpl.)

\Glsxtrheadshortpl{⟨entry-label⟩}

The header plural sentence case short form. (The behaviour of \Glsfmtshortpl when it occurs in a header.)

\GLSxtrtitleshortpl{⟨entry-label⟩}

The title plural all caps short form. (Normal behaviour of \GLSfmtshortpl.)

\GLSxtrheadshortpl{⟨entry-label⟩}

The header plural all caps short form. (The behaviour of \GLSfmtshortpl when it occurs in a header.)

\glsxtrtitlelong{⟨entry-label⟩}

The title long form. (Normal behaviour of \glsfmtlong.)

\glsxtrheadlong{⟨entry-label⟩}

The header long form. (The behaviour of \glsfmtlong when it occurs in a header.)

\Glsxtrtitlelong{⟨entry-label⟩}

The title sentence case long form. (Normal behaviour of \Glsfmtlong.)

\Glsxtrheadlong{⟨entry-label⟩}

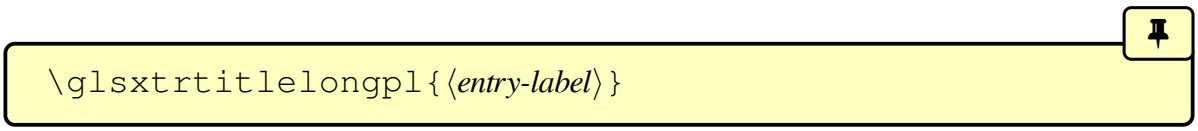
The header sentence case long form. (The behaviour of \Glsfmtlong when it occurs in a header.)

\GLSxtrtitlelong{⟨entry-label⟩}

The title all caps long form. (Normal behaviour of \GLSfmtlong.)

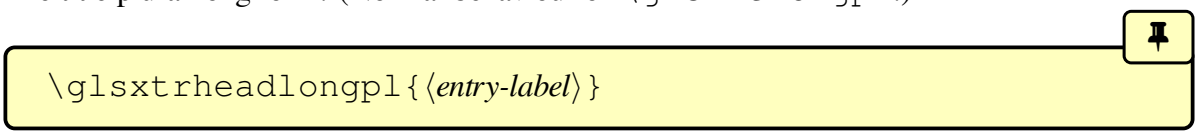
\GLSxtrheadlong{⟨entry-label⟩}

The header all caps long form. (The behaviour of \GLSfmtlong when it occurs in a header.)



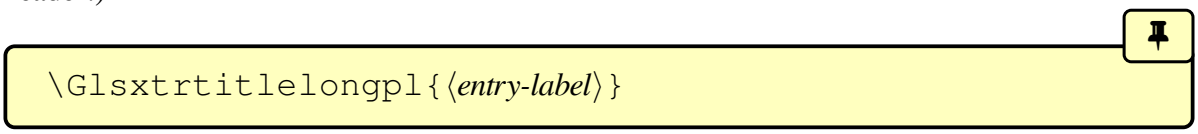
\glsxtrtitlelongpl{⟨entry-label⟩}

The title plural long form. (Normal behaviour of \glsfmtlongpl.)



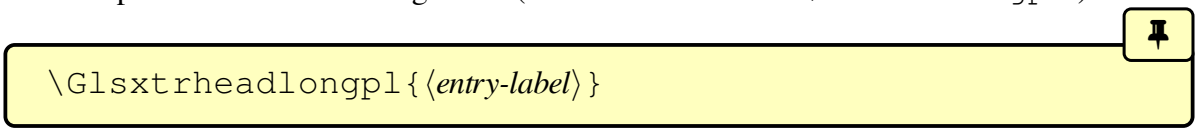
\glsxtrheadlongpl{⟨entry-label⟩}

The header plural long form. (The behaviour of \glsfmtlongpl when it occurs in a header.)



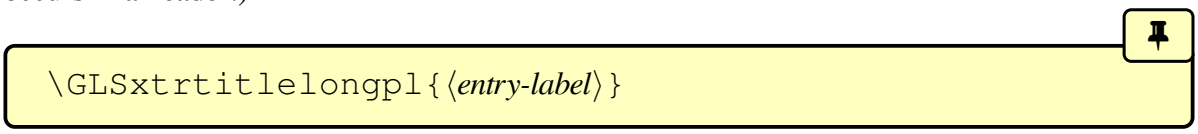
\Glsxtrtitlelongpl{⟨entry-label⟩}

The title plural sentence case long form. (Normal behaviour of \Glsfmtlongpl.)



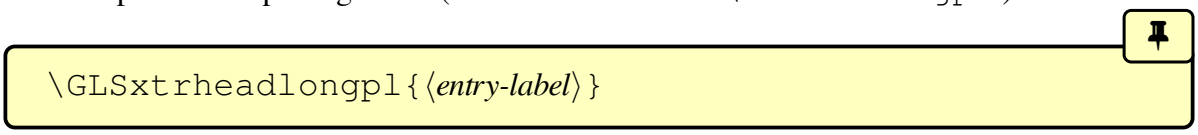
\Glsxtrheadlongpl{⟨entry-label⟩}

The header plural sentence case long form. (The behaviour of \Glsfmtlongpl when it occurs in a header.)



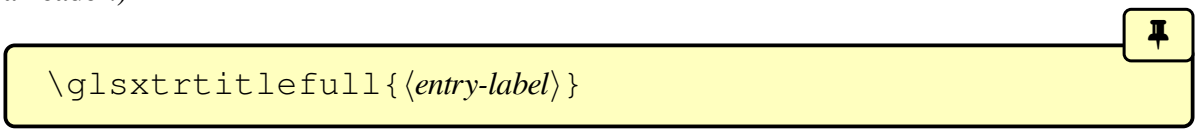
\GLSxtrtitlelongpl{⟨entry-label⟩}

The title plural all caps long form. (Normal behaviour of \GLSfmtlongpl.)



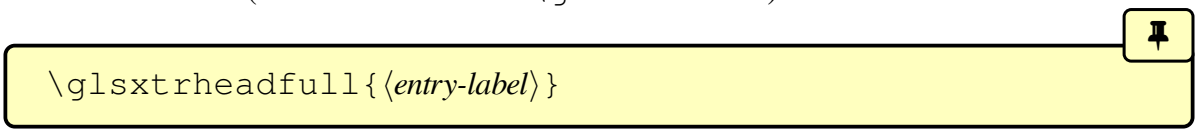
\GLSxtrheadlongpl{⟨entry-label⟩}

The header plural all caps long form. (The behaviour of \GLSfmtlongpl when it occurs in a header.)



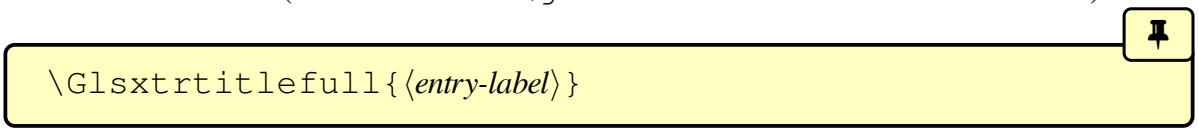
\glsxtrtitlefull{⟨entry-label⟩}

The title full form. (Normal behaviour of \glsfmtfull.)



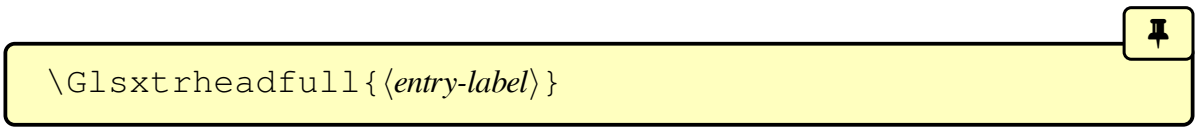
\glsxtrheadfull{⟨entry-label⟩}

The header full form. (The behaviour of \glsfmtfull when it occurs in a header.)



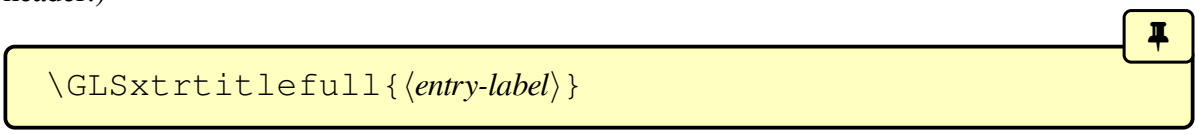
\Glsxtrtitlefull{⟨entry-label⟩}

The title sentence case full form. (Normal behaviour of \Glsfmtfull.)



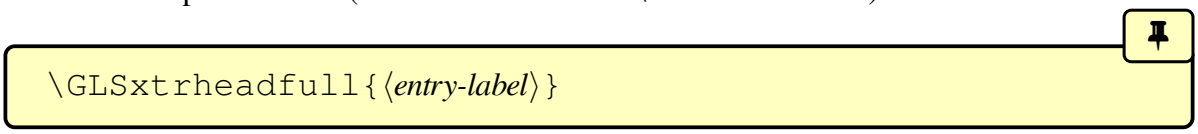
\Glsxtrheadfull{⟨entry-label⟩}

The header sentence case full form. (The behaviour of \Glsfmtfull when it occurs in a header.)



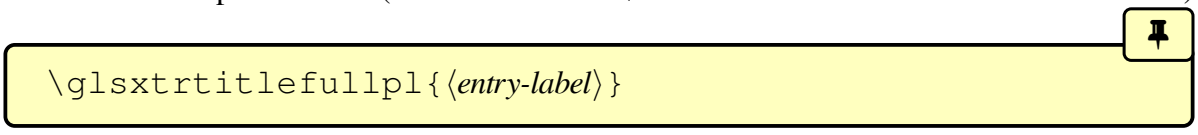
\GLSxtrtitlefull{⟨entry-label⟩}

The title all caps full form. (Normal behaviour of \GLSfmtfull.)



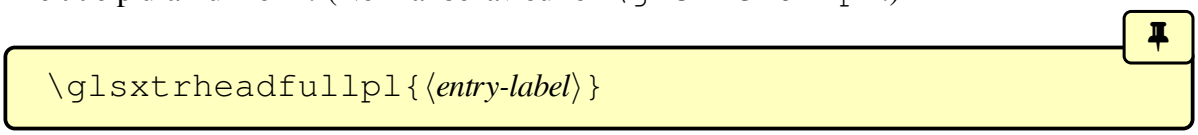
\GLSxtrheadfull{⟨entry-label⟩}

The header all caps full form. (The behaviour of \GLSfmtfull when it occurs in a header.)



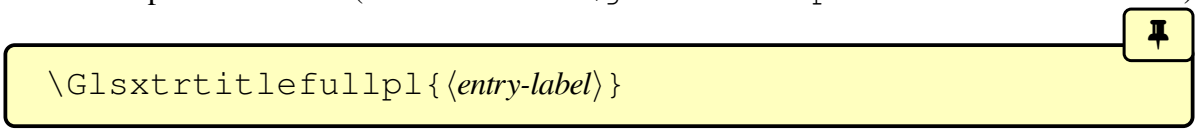
\glsxtrtitlefullpl{⟨entry-label⟩}

The title plural full form. (Normal behaviour of \glsfmtfullpl.)



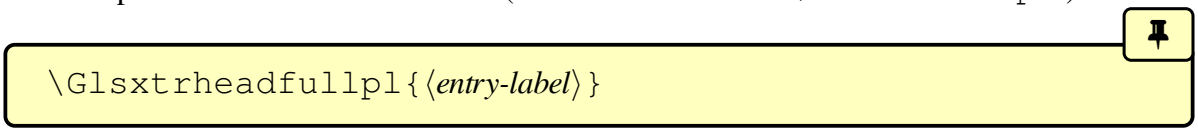
\glsxtrheadfullpl{⟨entry-label⟩}

The header plural full form. (The behaviour of \glsfmtfullpl when it occurs in a header.)



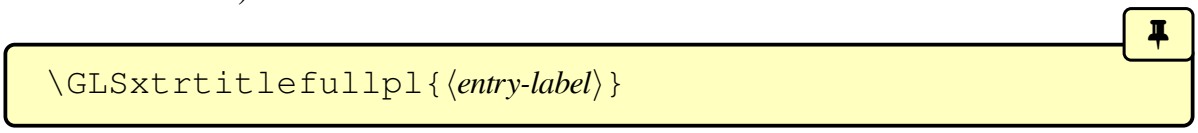
\Glsxtrtitlefullpl{⟨entry-label⟩}

The title plural sentence case full form. (Normal behaviour of \Glsfmtfullpl.)



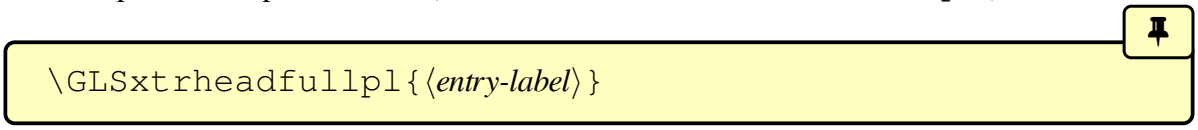
\Glsxtrheadfullpl{⟨entry-label⟩}

The header plural sentence case full form. (The behaviour of \Glsfmtfullpl when it occurs in a header.)



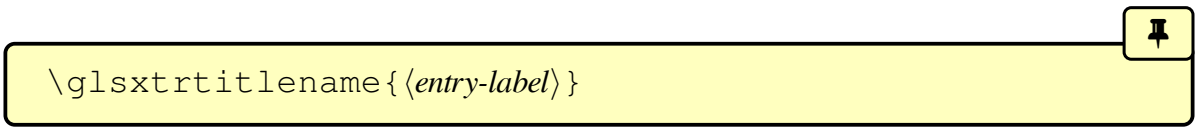
\GLSxtrtitlefullpl{⟨entry-label⟩}

The title plural all caps full form. (Normal behaviour of \GLSfmtfullpl.)



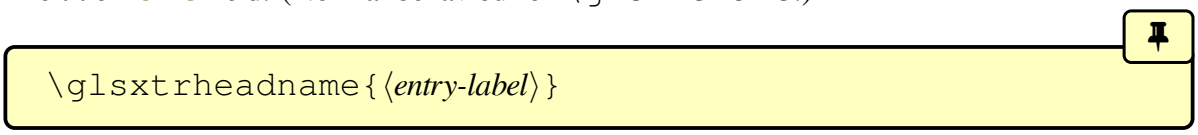
\GLSxtrheadfullpl{⟨entry-label⟩}

The header plural all caps full form. (The behaviour of \GLSfmtfullpl when it occurs in a header.)



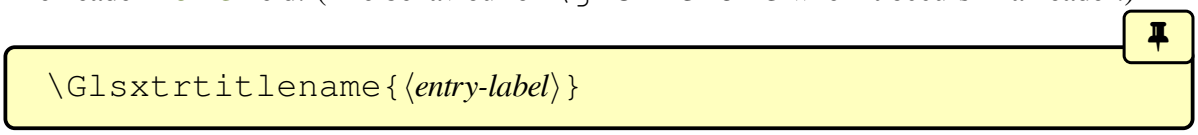
\glsxtrtitlename{⟨entry-label⟩}

The title name field. (Normal behaviour of \glsfmtname.)



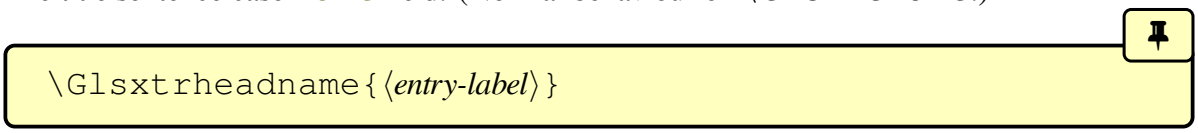
\glsxtrheadname{⟨entry-label⟩}

The header name field. (The behaviour of \glsfmtname when it occurs in a header.)



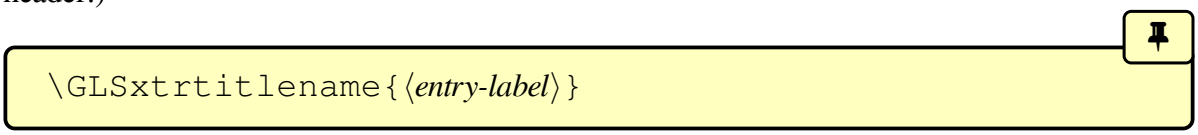
\Glsxtrtitlename{⟨entry-label⟩}

The title sentence case name field. (Normal behaviour of \Glsfmtname.)



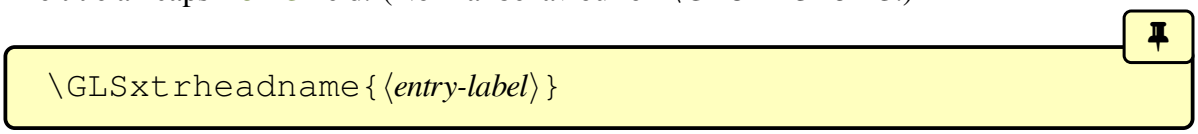
\Glsxtrheadname{⟨entry-label⟩}

The header sentence case name field. (The behaviour of \Glsfmtname when it occurs in a header.)



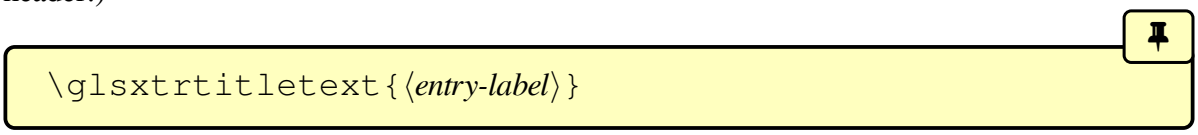
\GLSxtrtitlename{⟨entry-label⟩}

The title all caps name field. (Normal behaviour of \GLSfmtname.)



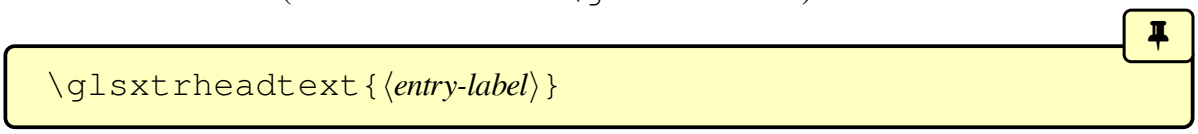
\GLSxtrheadname{⟨entry-label⟩}

The header all caps name field. (The behaviour of \GLSfmtname when it occurs in a header.)



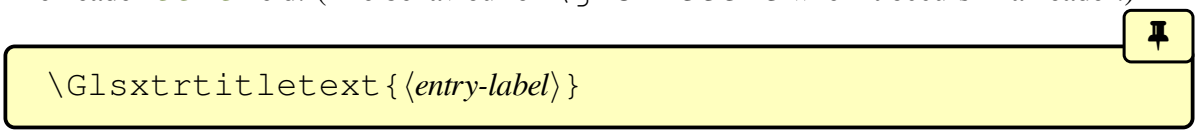
\glsxtrtitletext{⟨entry-label⟩}

The title text field. (Normal behaviour of \glsfmttext.)



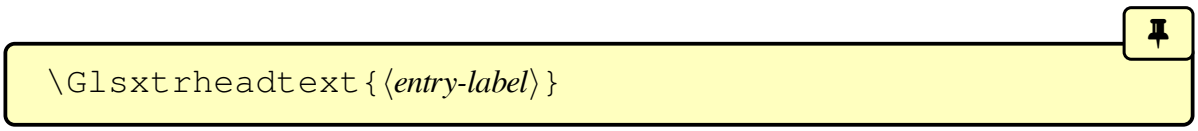
\glsxtrheadtext{⟨entry-label⟩}

The header text field. (The behaviour of \glsfmttext when it occurs in a header.)



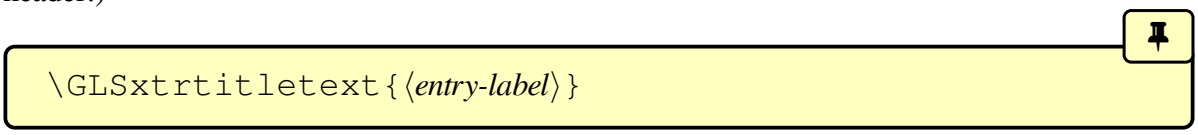
\Glsxtrtitletext{⟨entry-label⟩}

The title sentence case text field. (Normal behaviour of \Glsfmttext.)



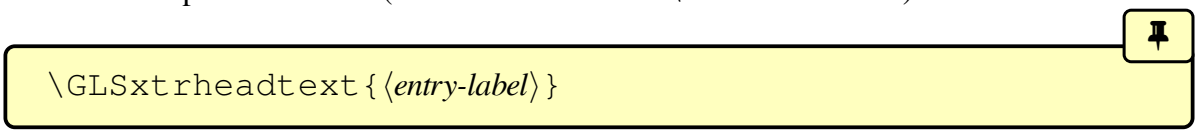
\Glsxtrheadtext{⟨entry-label⟩}

The header sentence case text field. (The behaviour of \Glsfmttext when it occurs in a header.)



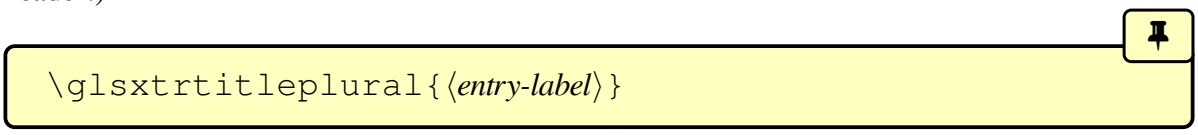
\GLSxtrtitletext{⟨entry-label⟩}

The title all caps text field. (Normal behaviour of \GLSfmttext.)



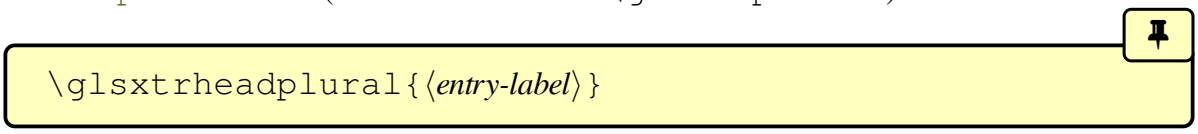
\GLSxtrheadtext{⟨entry-label⟩}

The header all caps text field. (The behaviour of \GLSfmttext when it occurs in a header.)



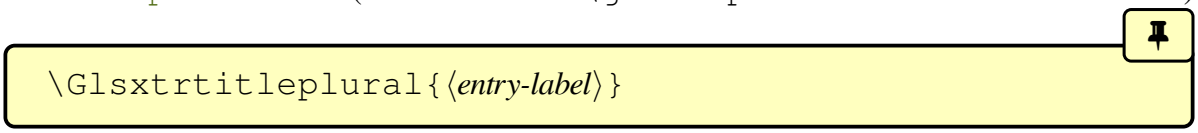
\glsxtrtitleplural{⟨entry-label⟩}

The title plural field. (Normal behaviour of \glsfmtplural.)



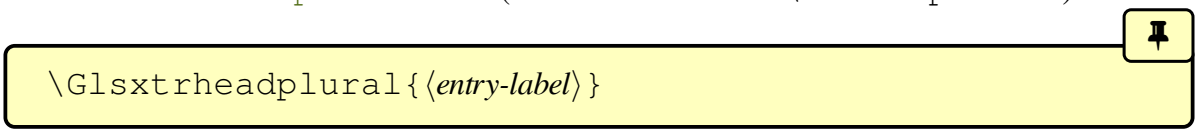
\glsxtrheadplural{⟨entry-label⟩}

The header plural field. (The behaviour of \glsfmtplural when it occurs in a header.)



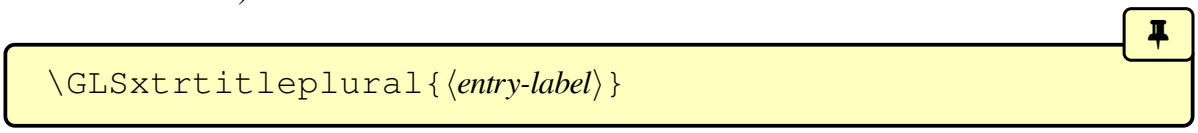
\Glsxtrtitleplural{⟨entry-label⟩}

The title sentence case plural field. (Normal behaviour of \Glsfmtplural.)



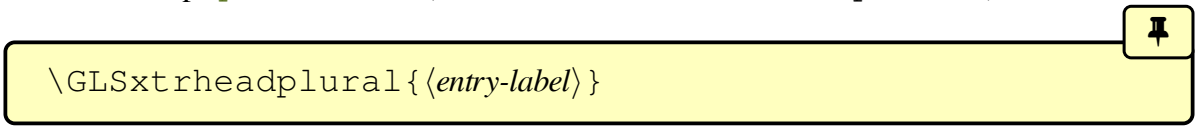
\Glsxtrheadplural{⟨entry-label⟩}

The header sentence case plural field. (The behaviour of \Glsfmtplural when it occurs in a header.)



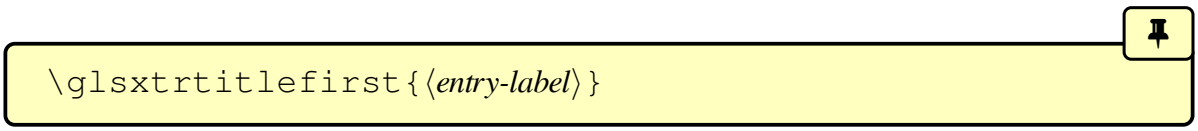
\GLSxtrtitleplural{⟨entry-label⟩}

The title all caps plural field. (Normal behaviour of \GLSfmtplural.)



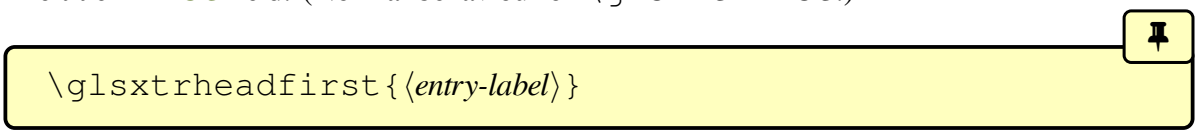
\GLSxtrheadplural{⟨entry-label⟩}

The header all caps plural field. (The behaviour of \GLSfmtplural when it occurs in a header.)



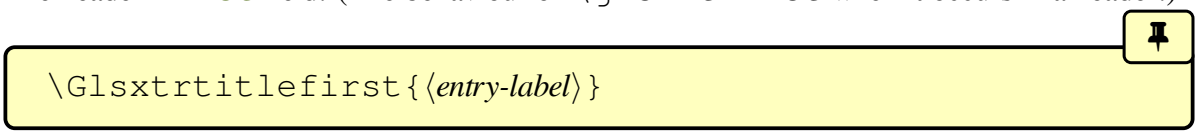
\glsxtrtitlefirst{⟨entry-label⟩}

The title first field. (Normal behaviour of \glsfmtfirst.)



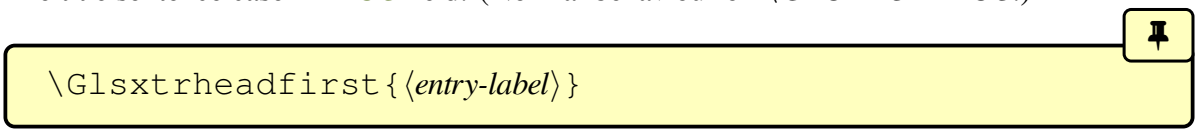
\glsxtrheadfirst{⟨entry-label⟩}

The header first field. (The behaviour of \glsfmtfirst when it occurs in a header.)



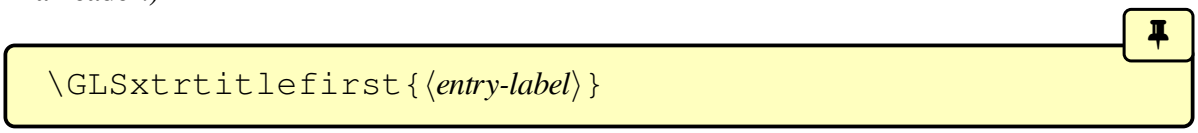
\Glsxtrtitlefirst{⟨entry-label⟩}

The title sentence case first field. (Normal behaviour of \Glsfmtfirst.)



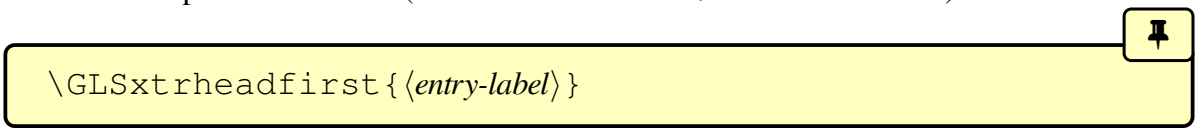
\Glsxtrheadfirst{⟨entry-label⟩}

The header sentence case first field. (The behaviour of \Glsfmtfirst when it occurs in a header.)



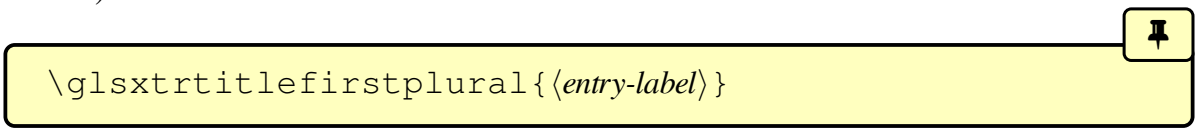
\GLSxtrtitlefirst{⟨entry-label⟩}

The title all caps first field. (Normal behaviour of \GLSfmtfirst.)



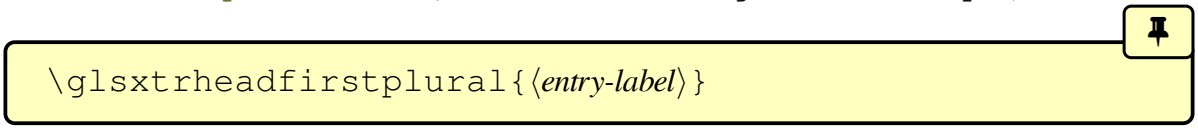
\GLSxtrheadfirst{⟨entry-label⟩}

The header all caps first field. (The behaviour of \GLSfmtfirst when it occurs in a header.)



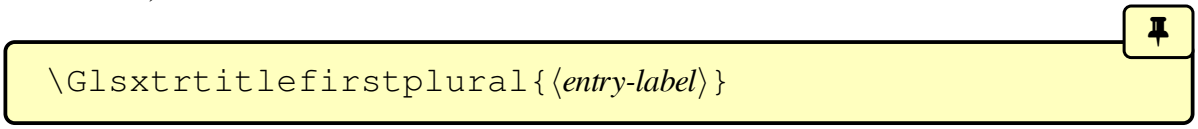
\glsxtrtitlefirstplural{⟨entry-label⟩}

The title firstplural field. (Normal behaviour of \glsfmtfirstpl.)



\glsxtrheadfirstplural{⟨entry-label⟩}

The header firstplural field. (The behaviour of \glsfmtfirstpl when it occurs in a header.)



\Glsxtrtitlefirstplural{⟨entry-label⟩}

The title sentence case firstplural field. (Normal behaviour of \Glsfmtfirstpl.)

\Glsxtrheadfirstplural{⟨*entry-label*⟩}

The header sentence case `firstplural` field. (The behaviour of `\Glsfmtfirstpl` when it occurs in a header.)

\GLSxtrtitlefirstplural{⟨*entry-label*⟩}

The title all caps `firstplural` field. (Normal behaviour of `\GLSfmtfirstpl`.)

\GLSxtrheadfirstplural{⟨*entry-label*⟩}

The header all caps `firstplural` field. (The behaviour of `\GLSfmtfirstpl` when it occurs in a header.)

The definitions of `\markright`, `\markboth` and `\@starttoc` are saved (using `\let`) when glossaries-extra loads.

\@glsxtr@org@markright{⟨*text*⟩}

The previous definition of `\markright`.

\@glsxtr@org@markboth{⟨*left text*⟩}{⟨*right text*⟩}

The previous definition of `\markboth`.

\@glsxtr@org@@starttoc{⟨*toc*⟩}

The previous definition of `\@starttoc`.

The glossaries-extra definitions of `\markright`, `\markboth` and `\@starttoc` all start and end with hooks that redefine commands that are sensitive to being in the header or contents.

\glsxtrmarkhook

This saves the original definitions and redefines the sensitive commands. This includes `\MakeUppercase` which is `\let` to `\MakeTextUppercase`.

\@glsxtrinmark

This redefines `\glsxtrifinmark` to just do its first argument (⟨*true*⟩).

```
\@glsxtrnotinmark
```

This redefines `\glsxtrifinmark` to just do its second argument (⟨*false*⟩).

```
\glsxtrrestoremarkhook
```

This restores the sensitive commands to the saved definitions. (For use where grouping will cause interference.) For example, `\markboth` is redefined as:

```
\renewcommand*{\markboth}[2]{%
 \glsxtrmarkhook
 \@glsxtr@org@markboth
   {\@glsxtrinmark#1\@glsxtrnotinmark}%
   {\@glsxtrinmark#2\@glsxtrnotinmark}%
 \glsxtrrestoremarkhook
}
```

5.4. Nested Links

Complications arise when you use the `\gls`-like commands in the value of the `name` field (or `text` or `first` fields, if set). This tends to occur with abbreviations that extend other abbreviations. For example, SHTML is an abbreviation for SSI enabled HTML, where SSI is an abbreviation for Server Side Includes and HTML is an abbreviation for Hypertext Markup Language.

For example, things can go wrong if the following is used with the glossaries package:

```
\newacronym{ssi}{SSI}{Server Side Includes}
\newacronym{html}{HTML}{Hypertext Markup Language}
\newacronym{shtml}{S\gls{html}}{\gls{ssi}
enabled \gls{html}}
```

The main problems are:

1. With older versions of mfirstuc and glossaries, the sentence case commands, such as `\Gls`, won't work for the `shtml` entry on first use if the long form is displayed before the short form (which is the default abbreviation style). This will attempt to do

   ```
   \gls{\uppercase ssi} enabled \gls{html}
   ```

which just doesn't work. Grouping the \gls{ssi} doesn't work either as this will effectively try to do:

```
\uppercase{\gls{ssi}} enabled \gls{html}
```

This will upper case the label ssi so the entry won't be recognised. This problem will also occur if you use the all caps version, such as \GLS{shtml}.

With mfirstuc v2.08+ and glossaries v1.49+, this issue should now be resolved for sentence case where \gls{ssi} will be mapped to \Gls{ssi} within \Gls{shtml}. The all caps command \GLS{shtml} will treat \gls as an exclusion and so won't perform a case-change. See §5.2 for further details.

2. The long and abbreviated forms accessed through \glsentrylong and \glsentryshort are no longer expandable and so can't be used be used in contexts that require this, such as PDF bookmarks.

3. The nested commands may end up in the sort key, which will confuse the indexing.

4. The shtml entry produces inconsistent results depending on whether the ssi or html entries have been used. Suppose both ssi and html are used before shtml. For example:

```
This section discusses
\gls{ssi}, \gls{html}
and
\gls{shtml}.
```

This section discusses server side includes (SSI), hypertext markup language (HTML) and SSI enabled HTML (SHTML).

In the above, the first use of the shtml entry produces "SSI enabled HTML (SHTML)".

Now let's suppose the html entry is used before the shtml but the ssi entry is used after the shtml entry, for example:

```
The sample files are
either \gls{html} or
\gls{shtml}, but let's
first discuss \gls
{ssi}.
```

The sample files are either hypertext markup language (HTML) or server side includes (SSI) enabled HTML (SHTML), but let's first discuss SSI.

In this case, the first use of the shtml entry now produces "server side includes (SSI) enabled HTML (SHTML)", which looks a bit cumbersome.

Now let's suppose the shtml entry is used before (or without) the other two entries:

```
This article is an
introduction
to \gls{shtml}.
```

This article is an introduction to server side includes (SSI) enabled hypertext markup language (HTML) (SHTML).

Now the first use of the `shtml` entry produces "server side includes (SSI) enabled hypertext markup language (HTML) (SHTML)", which looks strange.

This is all aggravated when using just the base glossaries package when the acronym style is set with `\setacronymstyle`. For example:

```
\setacronymstyle{long-short}
```

as this references the label through the use of `\glslabel` when displaying the long and short forms, but this value changes with each use of `\gls`, so instead of displaying "(SHTML)" at the end of the first use, it now displays "(HTML)", since `\glslabel` has been changed to `html` by `\gls{html}`.

In v1.48, the glossaries-extra package added grouping with `\glslinkwrcontent`, which scoped the link text. Unfortunately this grouping caused problems in math mode and had to be removed in v1.49. You can redefine `\glslinkwrcontent` to put the grouping back, but it still won't scope the definitions of the placeholder commands, such as `\glslabel`, which need to be outside of this scope for the benefit of the post-link hook.

Another oddity occurs if you reset the `html` entry between uses of the `shtml` entry. For example:

```
\gls{shtml} ... \glsreset{html}\gls{shtml}
```

The next use of `shtml` produces "Shypertext markup language (HTML)", which is downright weird. (This is a result of the short form being set to `S\gls{html}`, but `\gls{html}` is showing the full form.)

Even without this, the short form has nested formatting commands, which amount to `\acronymfont{S\acronymfont{HTML}}`. This may not be a problem for some styles, but if you use one of the "sm" styles (that use `\textsmaller`), this will produce an odd result.

5. Each time the `shtml` entry is used, the `html` entry will also be indexed and marked as used, and on first use this will happen to both the `ssi` and `html` entries. This kind of duplication in the location list isn't usually particularly helpful to the reader.

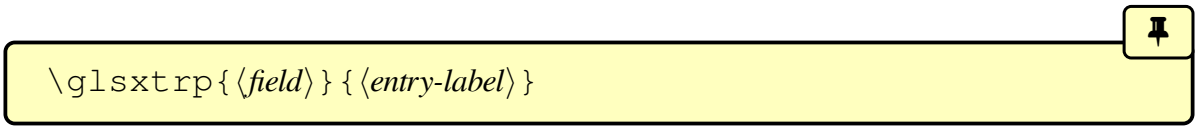
6. If hyperref is in use, you'll get nested hyperlinks and there's no consistent way of dealing with this across the available PDF viewers. If on the first use case, the user clicks on the "HTML" part of the "SSI enabled HTML (SHTML)" link, they may be directed to the HTML entry in the glossary or they may be directed to the SHTML entry in the glossary.

For these reasons, with just the base glossaries package, it's better to use the simple expandable commands like `\glsentrytext` or `\glsentryshort` in the definition of other entries.

The glossaries−extra package provides two other ways of dealing with these problems:

1. If the term can simply be treated as a series of previously defined entries, then consider using multi-entries (or compound sets), as described in §7. This deals with all the issues, including case-changing.

2. Use the partially-expandable `\glsxtrp`, described below.

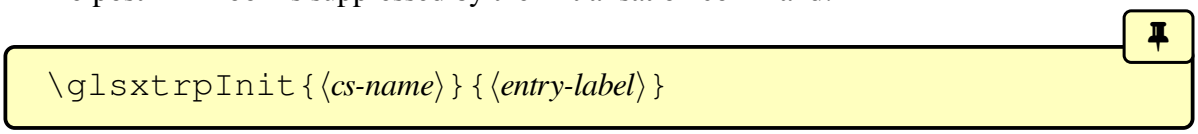


where ⟨*field*⟩ is the internal field label.

This command partially expands, so it will expand to just the field value if it occurs in the PDF bookmarks. Otherwise it will behave much like the commands described in §5.3.2, but with additional outer scoping and the post-link hook is suppressed.

Rather than testing the existence of the given field, this tests the existence of `\gls`⟨*field*⟩ or `\glsxtr`⟨*field*⟩, which means that it may be confused if the ⟨*field*⟩ argument is set to something that isn't a field but happens to match either of those command names (such as `full`).

The post-link hook is suppressed by the initialisation command:



This is used inside the added outer scoping and is simply defined as:

```
\newcommand\glsxtrpInit[2]{%
 \let\glspostlinkhook\relax}
```

It is possible to redefine this command to allow the `\glspostlinkhook` to be used, but any look-ahead (such as checking for a following punctuation character) won't work because of the added grouping. The arguments are ignored by default. If you want to redefine `\glsxtrpInit` the first argument is the name of the control sequence that will be used, without the leading backslash (for example, `glstext` or `glsxtrshort`) and the second argument is the entry's label.

Note that, as with commands like `\glsfmtshort`, there's no optional argument. The default settings are `noindex` and `hyper=false`. You can change this with:

```
\glsxtrsetpopts{⟨options⟩}
```

The argument should be the new default options.

At the start of each glossary, the default options are locally changed with:

```
\glossxtrsetpopts
```

This is defined as:

```
\newcommand*{\glossxtrsetpopts}{%
  \glsxtrsetpopts{noindex}%
}
```

This allows hyperlinks for any instance of `\glsxtrp` that occurs in the name or description, where it shouldn't be problematic.

There are also sentence case and all caps versions.

```
\Glsxtrp{⟨field⟩}{⟨entry-label⟩}
```

This uses the corresponding sentence case command `\Gls`⟨*field*⟩ or `\Glsxtr`⟨*field*⟩.

```
\GLSxtrp{⟨field⟩}{⟨entry-label⟩}
```

This uses the corresponding all caps command `\GLS`⟨*field*⟩ or `\GLSxtr`⟨*field*⟩.

There are some shortcut commands for the most common fields:

```
\glsps{⟨entry-label⟩}
```

which is equivalent to `\glsxtrp{short}{`⟨*entry-label*⟩`}`, and

```
\glspt{⟨entry-label⟩}
```

which is equivalent to `\glsxtrp{text}{`⟨*entry-label*⟩`}`. As well as sentence case and all caps versions:

```
\Glsps{⟨entry-label⟩}
```

which is equivalent to `\Glsxtrp{short}{`⟨*entry-label*⟩`}`,

```
\Glspt{⟨entry-label⟩}
```

which is equivalent to `\Glsxtrp{text}{⟨entry-label⟩}`,

```
\GLSps{⟨entry-label⟩}
```

which is equivalent to `\GLSxtrp{short}{⟨entry-label⟩}`, and

```
\GLSpt{⟨entry-label⟩}
```

which is equivalent to `\GLSxtrp{text}{⟨entry-label⟩}`.

Example 96 uses `\glsps` in the long form:

96

```
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-em-short-em}
\newabbreviation{html}{HTML}
{hypertext markup language}
\newabbreviation{ssi}{SSI}{server-side includes}
\newabbreviation{shtml}{SHTML}{\glsps{html} enabled
\glsps{ssi}}
\begin{document}
\tableofcontents
\section{\Glsfmtlong{shtml}}
First use: \gls{shtml}, \gls{html}, \gls{ssi}.

Next use: \gls{shtml}, \gls{html}, \gls{ssi}.

\printunsrtglossaries
\end{document}
```

(For a `bib2gls` alternative, see Example 152.) Note that the nested HTML and SSI are upright not italic. This is because `\emph` toggles between italic and upright, so the nested `\emph` switches back to upright. The emphasized style long-em-short-em was used to demonstrate this.

The way that this works is as follows:

- `\glsfmtlong{shtml}` expands to `\glsentrylong{shtml}` within the PDF bookmarks, which expands to the value of the `long` field:

Example 96: Nested link text with `\glspl`

Contents

1 HTML *enabled* SSI

First use: HTML *enabled* SSI (*SHTML*), *hypertext markup language* (*HTML*), *server-side includes* (*SSI*).

Next use: *SHTML*, *HTML*, *SSI*.

Glossary

HTML *hypertext markup language*

SSI *server-side includes*

SHTML HTML *enabled* SSI

```
\glsps{html} enabled \glsps{ssi}
```

This means that \glsps (within the PDF bookmarks) in turn expands to \glsentryshort. So the bookmark text (which can't contain any formatting commands) ends up as "HTML enabled SSI". The sentence case \Glsfmtlong{shtml} expands to \glspdfsentencecase{\glsentrylong{shtml}} within the PDF bookmarks. This expands the value of the long field before applying the case-change. However, \glsps only expands as far as \glsxtrp and no further. This means that \glspdfsentencecase has no effect as \glsxtrp is a case-exclusion command.

- \glsfmtlong{shtml} essentially behaves like \glsxtrlong, but with the indexing and hyperlink suppressed. The link text is the value of the long field encapsulated with the abbreviation style's formatting command (\glslongemfont in this case):

  ```
  \glslongemfont{\glsps{html} enabled \glsps{ssi}}
  ```

  This then becomes:

  ```
  \glslongemfont{{\let\glspostlinkhook\relax
   \glsxtrshort{html}} enabled
  {\let\glspostlinkhook\relax
   \glsxtrshort{ssi}}}
  ```

  Note the grouping and localised suppression of the post-link hook. The sentence case \Glsfmtlong similarly behaves like \Glsxtrlong, again with the indexing and hyperlink suppressed. In this example, there's no noticeable difference between using \glsfmtlong and \Glsfmtlong.

Note that in the above example, with older versions of mfirstuc and glossaries, it's not possible to use \Glsxtrlong{shtml} or similar. The problem here is that it will attempt to do:

```
\makefirstuc{\glsps{html} enabled \glsps{ssi}}
```

This will essentially end up as:

```
\glsps{\uppercase html} enabled \glsps{ssi}
```

which doesn't work. If you want to protect against automated case-changes, such as using the glossdesc attribute, insert an empty brace at the start:

```
\newabbreviation{shtml}{SHTML}{{}\glsps{html}
enabled \glsps{ssi}}
```

Alternatively, upgrade to mfirstuc v2.08+ and glossaries v4.50+. See §5.2.

5.5. Adjusting the Text Style

The `\gls`-like and `\glstext`-like commands produce text that's essentially formatted either as (`hyperoutside=true`):

⟨*hyper-cs*⟩ { ⟨*target*⟩ } { ⟨*textformat-cs*⟩ { ⟨*content*⟩ } } ⟨*post-link hook*⟩

or (`hyperoutside=false`):

⟨*textformat-cs*⟩ { ⟨*hyper-cs*⟩ { ⟨*target*⟩ } { ⟨*content*⟩ } } ⟨*post-link hook*⟩

If hyperlinks are enabled then ⟨*hyper-cs*⟩ creates the hyperlink based on ⟨*target*⟩ with the hyperlink text given by the second argument. If hyperlinks aren't enabled then ⟨*hyper-cs*⟩ ignores the ⟨*target*⟩ argument and simply does the second argument.

The ⟨*content*⟩ part is the link text, which includes the final optional ⟨*insert*⟩ (if supplied). The actual content depends on the command used (for example, `\gls` or `\glstext`). The `\gls`-like commands all use the entry display style associated with the entry's glossary type, (see §5.5.5). The `\glstext`-like commands set the ⟨*content*⟩ to the corresponding field value with the insert appended, all encapsulated with the inner formatting (see §5.5.3), with appropriate case-changing, if required.

The abbreviation commands (`\glsxtrshort`, `\glsxtrlong`, `\glsxtrfull` etc) are considered part of the set of `\glstext`-like commands, but the content is set according to the abbreviation style (see §4.5).

The commands `\glsdisp` and `\glslink` both have the content part explicitly set in their final argument. There's no insert optional argument as it can simply be included in the content part. The difference between them is that `\glsdisp` is considered a `\gls`-like command (it unsets the first use flag, §5.10, and uses the entry display style, §5.5.5), whereas `\glslink` is considered a `\glstext`-like command.

The ⟨*post-link hook*⟩ part is described in §5.5.4.

The ⟨*textformat-cs*⟩ command is the *outer* formatting command, described in §5.5.1. This doesn't include the post-link hook. If you want to include the post-link hook then you need to encapsulate the entire `\gls`-like and `\glstext`-like command (including the final optional argument, if present, and following punctuation, if the post-link hook looks ahead for punctuation).

Some sensitive formatting commands need to have the actual text in their argument (or else have the argument in an unbreakable box). The ⟨*content*⟩ part is usually too complicated for these commands. To help support this type of command, there is also an inner format, which is described in §5.5.3. In general, unless you require one of these sensitive commands, avoid setting

the inner text format as it requires support from the underlying style (either the entry format style or the abbreviation style), which may not be available.

Example 97 is ugly, but demonstrates the outer formatting (`typewriter` font), middle formatting (**bold** for regular entries and *italic* for abbreviations), inner formatting (highlighted in yellow), hyperlinks (red), and the category post-link hook (the description follows in parentheses for general entries on first use). 97

```
\usepackage{courier}
\usepackage[T1]{fontenc}
\usepackage{xcolor}
\usepackage{soul}
\usepackage[colorlinks]{hyperref}
\usepackage[nogroupskip]{glossaries-extra}
% outer formatting:
\renewcommand{\glstextformat}[1]{\texttt{#1}}
% middle formatting:
\renewcommand{\glsxtrregularfont}[1]{\textbf{#1}}
\renewcommand{\glsxtrabbreviationfont}[1]{\textit
{#1}}
% inner formatting:
\renewcommand{\glsxtrdefaultentrytextfmt}[1]{%
 \hl{#1}}
% post-link hook for 'general' category:
\glsdefpostlink{general}{%
 \glsxtrpostlinkAddDescOnFirstUse}
% define entries:
\newglossaryentry{sample}{name={sample},description=
{an example}}
\newabbreviation{html}{HTML}
{hypertext markup language}
\newacronym{nasa}{NASA}
{National Aeronautics and Space Administration}
\begin{document}
First use: \gls{sample}, \gls{html}, \gls{nasa}.

Next use: \gls{sample}, \gls{html}, \gls{nasa}.
\printunsrtglossaries
\end{document}
```

Example 97: Link text styles: outer, middle, inner, hyperlinks and post-link hook

First use: **sample** (an example), *hypertext markup language (HTML)*, **NASA**.

Next use: **sample**, *HTML*, **NASA**.

Glossary

sample an example

HTML hypertext markup language

NASA National Aeronautics and Space Administration

Note that the hyperlink, outer and middle formatting aren't applied to the post-link hook. The `acronym` category has the short−nolong abbreviation style, which sets the `regular` attribute to true. This means that the NASA entry uses the regular middle format (`\glsxtrregularfont`) not the abbreviation middle format (`\glsxtrabbreviationfont`).

If you have a formatting command that needs to have its argument fully-expanded before being applied, you may be able to use:

```
\GlsXtrExpandedFmt{⟨cs⟩}{⟨content⟩}
```

This fully-expands ⟨*content*⟩ and does ⟨*cs*⟩{⟨*expanded-content*⟩}, where ⟨*cs*⟩ is a command that takes a single argument. For example, to use soul's underlining command `\ul`:

```
\renewcommand{\glsxtrregularfont}[1]{%
 \GlsXtrExpandedFmt{#1}}
```

(See Example 118.) This isn't guaranteed to work as the link text may contain fragile content.

The inner formatting can be unpredictable. For example, abbreviation styles are complicated and so the inner formatting command is included in some of the field values, such as the `name`, which is why the abbreviation name is highlighted in the glossary. In the above example, the inner formatting is included in the category post-link hook, but only because `\glsxtrpostlinkAddDescOnFirstUse` is designed to include it. If the category post-link hook was simply defined as:

```
\glsdefpostlink{general}{%
 \glsxtrifwasfirstuse{ (\glsentrydesc{\glslabel})}{}
}
```

then the inner formatting won't be applied, since it's not included in the hook.

Example 98 demonstrates this with a slightly modified version of the above Example 97 document (initial part of preamble that deals with loading packages and redefining formatting commands as before): 98

```
% post-link hook for 'general' category:
\glsdefpostlink{general}{%
 \glsxtrifwasfirstuse
   { (\glsentrydesc{\glslabel})}{}}
% style sets the post-link hook for 'abbreviation' category:
\setabbreviationstyle{long-postshort-user}
% style sets the post-link hook for 'acronym' category:
\setabbreviationstyle[acronym]{short-postfootnote}
% define entries:
\newglossaryentry{sample}{name={sample},description=
{an example}}
\newabbreviation{html}{HTML}{hypertext markup language}
\newacronym{nasa}{NASA}
{National Aeronautics and Space Administration}
\begin{document}
First use: \gls{sample}, \gls{html}, \gls{nasa}.

Next use: \gls{sample}, \gls{html}, \gls{nasa}.
\printunsrtglossaries
\end{document}
```

The "post" abbreviation styles put some content into the post-link hook and provide support for the inner formatting. The above example sets the abbreviation style to long-postshort-user. This sets up the post-link hook for the associated category (abbreviation, in this case) to show the parenthetical material. Be aware that this will override any previous definition of that hook. This style supports the inner formatting (so the parenthetical material is highlighted).

Similarly, the short-postfootnote style is applied to the acronym category, and sets the post-link hook for that category (which looks head for punctuation). The inner formatting is applied to the footnote text but not the marker.

The post-link hook for the general category is now much simpler and doesn't include support for the inner formatting, so it's not highlighted.

None of the post-link content is incorporated into the hyperlink, outer or middle formatting.

In general, it's better to adjust the abbreviation's style commands (see §4.5.1.3) rather than use the middle or inner formatting if abbreviations need to be displayed in a particular font.

Example 98: Link text styles: outer, middle, inner, hyperlinks and post-link hooks (custom and abbreviation style)

First use: **sample** (an example), *hypertext markup language* (HTML), *NASA*.[1]
Next use: **sample**, *HTML*, *NASA*.

Glossary

sample an example

HTML hypertext markup language

NASA National Aeronautics and Space Administration

[1] National Aeronautics and Space Administration

5.5.1. Outer Formatting

By default, the outer formatting is produced with `\glstextformat`, which is defined by the base glossaries package. However it can be replaced by the `textformat` category attribute or by the `textformat` option. The order of precedence (not cumulative) is: the option supplied to the `\gls`-like or `\glstext`-like command, the category attribute, `\glstextformat`.

Example 99 demonstrates this:

99

```
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\glsdefpostlink{general}
 { (\glsentrydesc{\glslabel})}
\newglossaryentry{sample}{name={sample},description=
{an example}}
\newcommand{\strong}[1]{\textbf{\color{green}#1}}
\renewcommand{\glstextformat}[1]{\emph{#1}}
\begin{document}
\gls{sample}[-insert].
\strong{\gls{sample}[-insert]}.

\glssetcategoryattribute{general}{textformat}
{strong}
\gls{sample}[-insert].
\gls[hyperoutside=false]{sample}[-insert].

\gls[textformat=textsf]{sample}[-insert].
\end{document}
```

Example 99: Changing the outer text format

sample-insert (an example). ***sample-insert*** **(an example)**.
sample-insert (an example). **sample-insert** (an example).
sample-insert (an example).

The red text colour is from the hyperlink (red is the default with hyperref's `colorlinks` option). The green from the custom `\strong` command is cancelled by the hyperlink colour change when the hyperlink is inside `\strong`.

After the `textformat` attribute is set, the `\glstextformat` command isn't used, which is why the remaining lines don't have any italic. The final line uses the `textformat` option, which overrides the `textformat` attribute, so neither `\glstextformat` nor the custom `\strong` are used.

Note that the only time that the post-link hook is included in the formatting is when the entire `\gls` command has been encapsulated.

5.5.2. Middle Formatting

The middle formatting comes between the outer formatting (§5.5.1 above) and the inner formatting (§5.5.3 below).

The middle formatting is implemented by the entry format style (§5.5.5) for the `\gls`-like commands or is initialised by `\glsxtrassignfieldfont` for the `\glstext`-like commands.

If you provide your own custom entry format style you will need to add support for the middle formatting, if required.

`\glsxtrregularfont{⟨text⟩}`

The command to use for regular entries. This is initialised to just do its argument.

`\glsxtrabbreviationfont{⟨text⟩}`

The command to use for abbreviations that considered non-regular entries.

Example 100 has a regular entry (sample), a regular abbreviation (radar, which uses short−nolong the default `acronym` style), and a non-regular abbreviation (HTML, which uses long−short the default `abbreviation` style): 100

```
\newglossaryentry{sample}{name={sample},description=
{an example}}
\newabbreviation{html}{HTML}
{hypertext markup language}
\newacronym{radar}{radar}
{radio detection and ranging}
\renewcommand{\glsxtrregularfont}[1]{\emph{#1}}
\renewcommand{\glsxtrabbreviationfont}[1]{%
 \textbf{#1}}
\begin{document}
\gls{sample}, \gls{html}, \gls{radar}.
\end{document}
```

Example 100: Middle formatting

sample, **hypertext markup language (HTML)**, *radar*.

Note that even though radar is an abbreviation, it's considered a regular entry because it uses a regular style.

\glsxtrassignfieldfont{⟨*entry-label*⟩}

This command is used by all the \glstext-like commands to initialise the internal command used to encapsulate the field value. This will either be set to \glsxtrregularfont (for regular entries) or \@firstofone otherwise.

Note that this doesn't use \glsxtrabbreviationfont as non-regular abbreviations are too complicated to work with \glstext, \glsfirst, \glsplural, \glsfirstplural or their case-changing variants. Instead, use the \gls-like commands or the abbreviation commands, such as \glsxtrshort.

5.5.3. Inner Formatting

If you want to format the link text, the best method is to either use the outer formatting or encapsulate the entire \gls-like or \glstext-like command, as described in §5.5.1. However, there are some sensitive commands that don't work if the command argument doesn't simply contain text.

Sometimes the issue may occur when the sensitive command that needs to encapsulate \gls doesn't like boolean variables being changed (which occurs when the first use flag is unset). If this is the case, you may want to consider buffering as an alternative (see §5.10.1).

For example, if the sample document Example 99 from §5.5.1 is adjusted to include the soul package and the following line is added to the document:

```
\gls[textformat=hl]{sample}
```

then the document build will fail with the error:

> ! Package soul Error: Reconstruction failed.

Once solution is to do the following instead:

```
\hl{\mbox{\gls{sample}}}
```

This will now work, but the box will prevent hyphenation, so it's only useful if the link text is short, such as a symbol. If the link text is long (such as a phrase or the first use of an abbreviation), this method can produce undesirable results with overfull or underfull lines.

The inner formatting is designed to provide a workaround, but it must be implemented deep within the entry style formatting. This means that if you provide your own custom style, you will need to add the appropriate commands if you want that style to support inner formatting. You may also need to switch to using `\MFUsentencecase` instead of `\makefirstuc` if any of the sentence case commands are required:

```
\renewcommand{\glssentencecase}[1]{\MFUsentencecase{#1}}
```

Although there's no guarantee that this will work for some particularly problematic formatting commands.

With the default entry style, the above example can be changed to:

```
\gls[innertextformat=hl]{sample}
```

The inner formatting may be split up in order to move them into the arguments of internal commands, such as those used for case-changing. This can result in unwanted side-effects.

Example 101 uses `\fbox` (which draws a frame around its argument) and soul's `\so` (which spaces out the letters): 101

```
% requires glossaries.sty v4.50+ and mfirstuc v2.08+
\renewcommand{\glssentencecase}[1]{\MFUsentencecase
{#1}}
\newacronym{radar}{radar}
{radio detection and ranging}
\begin{document}
\Gls[innertextformat=fbox]{radar}['s] system\ldots

\Gls[innertextformat=so]{radar}['s] system\ldots

\fbox{\Gls{radar}['s]} system\ldots

\so{\mbox{\Gls{radar}['s]}} system\ldots
\end{document}
```

Example 101: Inner formatting

Radar 's system...
R a d a r' s system...
Radar's system...
Radar's system...

Note the fragmentation of the inner formatting. The use of \mbox in the final line prevents an error but the letters aren't spaced out. The only way to deal with this case is to use \glsdisp or \glslink with the text explicitly written:

```
\glslink{radar}{\so{Radar's}} system\ldots
```

The above example requires mfirstuc v2.08+.

Below are the commands used to support inner formatting.

\glsxtrgenentrytextfmt

This is the command that's used to encapsulate any content that should have the inner formatting applied. It should not be redefined within the document as it's initialised within the \gls-like and \glstext-like commands. It's used within \glsgenentryfmt and included in the helper commands used by the predefined abbreviation styles.

Sometimes it may be necessary to include \glsxtrgenentrytextfmt within the actual field value to ensure that it's as close as possible to the text. This is performed automatically when an entry is defined if the encapinnerfmt or encapnocaseinnerfmt attributes are set. Note that even in this case, fragmentation will occur with sentence case commands like \Gls or with the insert optional argument, as in the above example with \fbox and \so.

\glsxtrdefaultentrytextfmt{⟨*text*⟩}

This is the default command that \glsxtrgenentrytextfmt will be \let to within the \gls-like and \glstext-like commands before their options are processed. This simply does its argument but may be redefined. (See Example 118.)

\glsxtrattrentrytextfmt{⟨*text*⟩}

This command applies formatting according to whether or not the innertextformat attribute is set. It isn't used by default as it should rarely be needed and increases complexity. However, if you would like to provide support for the innertextformat attribute, you can redefine \glsxtrdefaultentrytextfmt to use \glsxtrattrentrytextfmt:

```
\renewcommand{\glsxtrdefaultentrytextfmt}{%
 \glsxtrattrentrytextfmt}
```

This command expects the entry label to be stored in \glslabel (from which it obtains the category label).

The \gls-like commands use \glsxtrgenentrytextfmt within \glsgenentryfmt for regular entries or within the abbreviation style commands for non-regular abbreviations (see §5.5.5).

The \glstext-like commands all essentially perform the following steps:

1. Initialise the middle formatting command ⟨*field-font-cs*⟩ used for encapsulating the field with \glsxtrassignfieldfont (see §5.5.2).

2. If \glsifapplyinnerfmtfield indicates that the field value should be encapsulated by \glsxtrgenentrytextfmt, then this essentially does (or appropriate case-change equivalent):

   ```
   ⟨field-font-cs⟩{\glsaccessfmt⟨field⟩{⟨insert⟩}%
    {\glsxtrgenentrytextfmt}{⟨entry-label⟩}{⟨internal-field⟩}}
   ```

   otherwise it does:

   ```
   ⟨field-font-cs⟩{\glsaccess⟨field⟩{⟨entry-label⟩}%
    \glsxtrgenentrytextfmt{⟨insert⟩}}
   ```

   (See §9 for the "access" commands.)

For example, the link text for \glstext is:

```
\glsifapplyinnerfmtfield{⟨entry-label⟩}{text}%
{%
 ⟨field-font-cs⟩{\glsaccessfmttext{⟨insert⟩}%
   {\glsxtrgenentrytextfmt}{⟨entry-label⟩}}%
}%
{%
 ⟨field-font-cs⟩{\glsaccesstext{⟨entry-label⟩}%
   \glsxtrgenentrytextfmt{⟨insert⟩}}%
}
```

The `\glsaccessfmt⟨field⟩` commands internally use `\glsfmtfield` to apply the inner formatting.

```
\glsifapplyinnerfmtfield{⟨entry-label⟩}{⟨internal-field⟩}{⟨true⟩}
{⟨false⟩}
```

This determines whether or not the field identified by its internal field label for the given entry should have its value encapsulated by the inner formatting command. False indicates that the field value already contains the inner formatting command.

```
\glsexclapplyinnerfmtfield{⟨entry-label⟩}{⟨internal-field⟩}
```

Locally adds the given field identified by its internal field label to the exclusion list for the given entry.

```
\glsfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

This command applies the formatting command ⟨*cs*⟩ (which takes one argument) to the entry's field value identified by the given internal field label, including ⟨*insert*⟩ appended. This ensures that the internal control sequence used to store the field's value is expanded before ⟨*cs*⟩ is applied.

```
\Glsfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

As above but sentence case.

```
\GLSfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

As above but all caps.

5.5.4. Post Link Hook

The post-link hook is a convenient way of automatically appending content after each instant of the \gls-like and \glstext-like commands. The simplest method of implementing this is with the category post-link hook, which is only applied to entries that have the given category.

Example 102 places an asterisk (*) after all entries with the default general category:

102

```
\glsdefpostlink{general}{*}
\newglossaryentry{sample}{name={sample},symbol={X},
 description={an example}}
\begin{document}
\Gls{sample}, \glstext{sample}, \glsdesc{sample}
and \glssymbol{sample}.
\end{document}
```

Example 102: Category post-link hook

Sample*, sample*, an example* and X*.

Typically, the category post-link hook is more likely to include some conditional, such as to only insert text on first use. For example, \glsxtrpostlinkAddDescOnFirstUse can be used to insert the description in parentheses after the first use.

The "post" abbreviation styles all set the category post-link hook, which will overwrite any previous definition for the abbreviation's category.

Within the post-link hook, you can use the placeholder commands, such as \glslabel (see §5.5.5), but note that you can't use \ifglsused to determine whether or not the entry has been used, since the post-link hook comes after the entry has been unset. Instead, use \glsxtrifwasfirstuse. Additional commands provided for use within the post-link hooks are described in this section.

The post-link hook is implemented with \glspostlinkhook, which is defined by the base glossaries package. It's used at the end of the \gls-like and \glstext-like commands. The original base definition does nothing, but glossaries-extra redefines this:

```
\renewcommand*{\glspostlinkhook}{%
 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
}
```

This uses:

\glsxtrpostlinkhook

which is the main glossaries-extra post-link hook.

If you are migrating over from only using the base glossaries package and you have redefined \glspostlinkhook, consider moving your modifications to the category post-link hook or prepend to \glsxtrpostlink, as some attributes and abbreviation styles rely on the features provided by \glsxtrpostlinkhook.

The main post-link hook is defined as:

```
\newcommand*{\glsxtrpostlinkhook}{%
 \glsxtrdiscardperiod{\glslabel}%
 {\glsxtrpostlinkendsentence}%
 {\glsxtrifcustomdiscardperiod
  {\glsxtrifperiod
    {\glsxtrpostlinkendsentence}%
    {\glsxtrpostlink}}%
  {\glsxtrpostlink}%
 }%
}
```

This checks if a following full stop needs to be discarded and does the inner post-link hook \glsxtrpostlink. Note that \glsxtrdiscardperiod and \glsxtrifperiod look ahead for a following token, so if you need to modify this command, insert your custom code at the start or add it to the category post-link hook instead.

\glsxtrdiscardperiod{⟨*entry-label*⟩}{⟨*discarded*⟩}{⟨*no discard*⟩}⟨*token*⟩

This discards ⟨*token*⟩ if it's a full stop and the entry's category attributes indicate that a full stop should be discarded (such as discardperiod). If the punctuation character is discarded, this will then do ⟨*discarded*⟩, otherwise it will do ⟨*no discard*⟩ and process ⟨*token*⟩ as usual. If the retainfirstuseperiod attribute is set, then the following command is used to determine whether or not to discard ⟨*token*⟩.

\glsxtrdiscardperiodretainfirstuse{⟨*entry-label*⟩}{⟨*discarded*⟩}{⟨*no discard*⟩}⟨*token*⟩

This was introduced in v1.49 and is defined as:

```
\newcommand*{\glsxtrdiscardperiodretainfirstuse}[3]
{%
 \glsxtrifwassubsequentorshort
   {\glsxtrifperiod{#2}{#3}}{#3}%
}
```

This will only discard the full stop if it follows the subsequent use of a `\gls`-like command or if it follows one of the `\glsxtrshort` set of commands. Note that this has a different effect from pre v1.49 with the `\glstext`-like commands, but it's more appropriate since it's typically only the short form that requires the period to be discarded. To restore the original behaviour:

```
\renewcommand*{\glsxtrdiscardperiodretainfirstuse}
[3]{%
 \glsxtrifwasfirstuse{#3}{\glsxtrifperiod{#2}{#3}}%
}
```

If you want your own custom code to determine whether or not to check for a period (instead of using known category attributes), you can redefine:

`\glsxtrifcustomdiscardperiod{⟨true⟩}{⟨false⟩}` *initial:* ⟨*false*⟩

This should expand to ⟨*true*⟩ if a check should be performed, otherwise it should expand to ⟨*false*⟩. The default definition simply does ⟨*false*⟩.

`\glsxtrpostlinkendsentence`

This is done if a full stop is discarded. If there is a category post-link hook for the entry's category, that hook is performed (`\glsxtrpostlink`⟨*category*⟩ not `\glsxtrpostlink`) and the full stop is put back followed by a space factor adjustment. Otherwise, just the space factor adjustment is done.

The test to determine whether or not ⟨*token*⟩ is a full stop is determined by:

`\glsxtrifperiod{⟨true⟩}{⟨false⟩}⟨token⟩`

It may be useful to test for other punctuation characters. For example, styles such as short−postfootnote will move the footnote after certain punctuation characters.

```
\glsxtrifnextpunc{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if it's followed by one of the set of recognised punctuation characters, otherwise it does false. The set is initialised to `.,:;?!` (full stop, comma, colon, semi-colon, question mark, and exclamation mark).

A convenient way of moving code after the punctuation character is to use:

```
\glsxtrdopostpunc{⟨code⟩}⟨token⟩
```

If ⟨*token*⟩ is a recognised punctuation character, this will place ⟨*code*⟩ after the token, otherwise it will be placed before the token.

Example 103 adapts Example 102 to put the asterisk after following punctuation:

103

```
\glsdefpostlink{general}{\glsxtrdopostpunc{*}}
\newglossaryentry{sample}{name={sample},symbol={X},
 description={an example}}
\begin{document}
\Gls{sample}, \glstext{sample},
(\glsdesc{sample}) and \glssymbol{sample}.
\end{document}
```

Example 103: Category post-link hook with punctuation lookahead

Sample,* sample,* (an example*) and X.*

Note that the asterisk isn't moved after the closing parenthesis. This is because that character isn't included in the default list.

You can add additional punctuation marks with:

```
\glsxtraddpunctuationmark{⟨token(s)⟩}
```

You may list multiple characters at the same time to add a batch, but don't add any separators (including spaces).

Note that each character must be a single token, which means a single-byte character for pdfLaTeX. Multi-byte characters (UTF-8) will required a native Unicode engine (XeLaTeX or LuaLaTeX).

For example:

```
\glsxtraddpunctuationmark{-'/}
```

This adds three extra punctuation marks (hyphen, apostrophe and slash). Note that this doesn't allow for closing double-quotes and will break `''` (double apostrophe sequence for a closing double-quote) if found. The following will only work with XeLaTeX or LuaLaTeX:

```
\usepackage{fontspec}
\usepackage{glossaries-extra}
\glsxtraddpunctuationmark{”}
```

You can set the list with:

```
\glsxtrsetpunctuationmarks{⟨token list⟩}
```

This will remove the default set as well as any additional characters. As above, each character must be a single token with no separators in the list. For example:

```
\glsxtrsetpunctuationmarks{.?!}
```

This sets the list to just three punctuation characters (so comma, colon, and semi-colon are no longer recognised).

```
\glsxtrpostlink
```

This does the category post-link hook (or nothing if it hasn't been defined):

```
\newcommand*{\glsxtrpostlink}{%
 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
}
```

Customisation is best performed within the category post-link hook, which can be defined (or redefined) with:

```
\glsdefpostlink{⟨category⟩}{⟨definition⟩}
```

The first argument is the category label and the second is the code to perform. Note that this doesn't check if the hook has already been defined for the category. The hook is a command in the form `\glsxtrpostlink⟨category⟩`. If the category label only consists of letters, you can also use `\newcommand` or `\renewcommand` instead.

\glspretopostlink{⟨category⟩}{⟨code⟩}

Similar to the above but prepends ⟨*code*⟩ to the associated hook (or simply defines it, if the hook doesn't already exist).

\glsapptopostlink{⟨category⟩}{⟨code⟩}

Similar to the above but appends ⟨*code*⟩ to the associated hook.

Take care not to choose category labels that will cause a conflict. For example, endsentence and hook will conflict with the commands \glsxtrpostlinkendsentence and \glsxtrpostlinkhook.

If you want code in the post-link hook that's not dependent on the category, consider prepending it to \glsxtrpostlink or \glsxtrpostlinkhook. Don't append it to \glsxtrpostlinkhook otherwise it will interfere with the punctuation lookahead.

For convenience, some commands are provided that may be of use in the category post-link hook:

\glsxtrpostlinkAddDescOnFirstUse

This will add the description in parentheses if the hook follows the first use of the entry. This incorporates the inner formatting and description accessibility support, if provided.

\glsxtrpostlinkAddSymbolOnFirstUse

This will add the symbol in parentheses if that field is set and the hook follows the first use of the entry. This incorporates the inner formatting and symbol accessibility support, if provided.

\glsxtrpostlinkAddSymbolDescOnFirstUse

This will add the symbol, if that field is set, and the description (both within the same set of parentheses), if the hook follows the first use of the entry. This incorporates the inner formatting and accessibility support, if provided. The separator between the symbol and description is given by:

\glsxtrpostlinkSymbolDescSep

The default is a comma followed by a space.

Example 104 defines post-link hooks for general, symbol and number categories:

104

```
\glsdefpostlink{general}
 {\glsxtrpostlinkAddDescOnFirstUse}
\glsdefpostlink{symbol}
 {\glsxtrpostlinkAddSymbolOnFirstUse}
\glsdefpostlink{number}
 {\glsxtrpostlinkAddSymbolDescOnFirstUse}
\newglossaryentry{sample}{name={sample},description=
{an example}}
\newglossaryentry{alpha}{name={alpha},symbol=
{\ensuremath{\alpha}},
 description={a symbol},category={symbol}}
\newglossaryentry{pi}{name={pi},symbol={\ensuremath
{\pi}},
 description={a constant},category={number}}
\begin{document}
First use: \gls{sample}, \gls{alpha}, \gls{pi}.

Next use: \gls{sample}, \gls{alpha}, \gls{pi}.
\end{document}
```

Example 104: Category post-link hooks

First use: sample (an example), alpha (α), pi (π, a constant).
Next use: sample, alpha, pi.

The following commands are also provided for use in the post-link hook:

```
\glsxtrcurrentfield
```

This expands to empty if the calling command isn't associated with one specific field (such as `\glslink`, the `\gls`-like commands, the inline full form commands) otherwise it will expand to the name of the key associated with the *singular* form of the command. For example, this command will expand to `text` for both `\glstext` and `\glsplural`, to `description` for both `\glsdesc` and `\glsdescplural`, and to `short` for both `\glsxtrshort` and `\glsxtrshortpl`. Whereas it will expand to nothing for both `\gls` and `\glsxtrfull`.

```
\glsxtrifwasglslike{⟨true⟩}{⟨false⟩}
```

This expands to ⟨*true*⟩ if the calling command was a `\gls`-like command and to ⟨*false*⟩ otherwise.

`\glsxtrifwasglslikeandfirstuse{⟨true⟩}{⟨false⟩}`

This expands to ⟨*true*⟩ if the calling command was a `\gls`-like command and was the first use otherwise it expands to ⟨*false*⟩. This is simply a shortcut command that uses both `\glsxtrifwasglslike` and `\glsxtrifwasfirstuse`.

`\glsxtrifwassubsequentuse{⟨true⟩}{⟨false⟩}`

This expands to ⟨*true*⟩ if the calling command was a `\gls`-like command and was the subsequent use otherwise it expands to ⟨*false*⟩. This is simply a shortcut command that uses both `\glsxtrifwasglslike` and `\glsxtrifwasfirstuse`.

`\glsxtrifwassubsequentorshort{⟨true⟩}{⟨false⟩}`

This expands to ⟨*true*⟩ if the calling command was a `\gls`-like command and was the subsequent use or if the calling command set `\glsxtrcurrentfield` to `short`. Otherwise it expands to ⟨*false*⟩.

`\glsxtrifallcaps{⟨all caps⟩}{⟨not all caps⟩}`

This simply does:

`\glscapscase{⟨not all caps⟩}{⟨not all caps⟩}{⟨all caps⟩}`

It's not usually necessary for the post-link hook to differentiate between no case-change and sentence case, so this provides a convenient shortcut if only the all caps case needs to be different.

It's possible you may also want to reference the inserted material. For the `\gls`-like commands, this can be obtained with the placeholder `\glsinsert`, but it's not normally set by the `\glstext`-like commands, which don't use the entry format style (§5.5.5) and instead incorporate the inserted material at the end of the link text. If you want the post-link hook to be able to access the inserted material for the `\glstext`-like commands, you must first save it, by redefining the following:

`\glsxtrsaveinsert{⟨entry-label⟩}{⟨insert⟩}`

This is used by the \glstext-like commands to initialise \glsinsert. The default is:

```
\newcommand*{\glsxtrsaveinsert}[2]{%
 \def\glsinsert{}}
```

For example, to always save the insert:

```
\renewcommand*{\glsxtrsaveinsert}[2]{%
 \def\glsinsert{#2}}
```

The first argument can be used to conditionally assign the insert. For example, the following will only save it for entries with the general category:

```
\renewcommand*{\glsxtrsaveinsert}[2]{%
 \glsifcategory{#1}{general}
  {\def\glsinsert{#2}}% general
  {\def\glsinsert{}}% not general
}
```

If you only want to save the insert for the \glsxtrfull set of commands, you can redefine \glsxtrfullsaveinsert instead (see §4.3).

```
\glsxtrassignlinktextfmt
```

This contains the assignments required to ensure that \glslabel, \glstextformat and \glsxtrgenentrytextfmt have the definitions they had within the link text. They would ordinarily still have those definitions within the post-link hook, but if, for example, the hook contains content that may be deferred, such as a footnote, then judicious use and expansion of \glsxtrassignlinktextfmt can allow the deferred code to pick up the label, outer and inner formatting.

For example, the post-link hook might contain:

```
\expandafter\footnote\expandafter
 {\glsxtrassignlinktextfmt\glstextformat{%
   \Glsaccessfmtdesc{}{\glsxtrgenentrytextfmt}
    {\glslabel}}.}
```

5.5.5. Entry Format Style

This section is for advanced users. Minor modifications can be made by adjusting the outer formatting (§5.5.1), the post-link hook (§5.5.4) or the abbreviation style commands (§4.5.1.3).

The \gls-like commands have the link text set to the entry format style corresponding to the entry's glossary type. This can be changed with \defglsentryfmt, but the default style is given by \glsentryfmt, which is defined by the base glossaries package. This uses the placeholder commands to determine the appropriate text. These are described in the glossaries manual, but to recap they are: \glslabel (the entry's label), \glscustomtext (text provided by \glsdisp or empty otherwise), \glsinsert (supplied in the final optional argument except for \glsdisp, empty by default), \glsifplural, and \glscapscase.

The glossaries-extra package redefines \glsentryfmt to test whether or not the entry is an abbreviation and, if so, whether or not the entry should be treated as a regular entry:

```
\renewcommand*{\glsentryfmt}{%
  \ifglshasshort{\glslabel}
    {\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
  \glsifregular{\glslabel}%
  {\glsxtrregularfont{\glsgenentryfmt}}%
  {%
    \ifglshasshort{\glslabel}%
    {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
    {\glsxtrregularfont{\glsgenentryfmt}}%
  }%
}
```

This uses \ifglshasshort to determine whether or not the entry is an abbreviation. If it is, then \glssetabbrvfmt is used to setup the abbreviation style commands for the entry's category.

Then there's a check (with \glsifregular) to determine whether or not the entry should be treated as a regular entry. Note that if the regular attribute hasn't been set to true, the entry will still be treated as a regular entry if it doesn't have the short field set.

Regular entries are formatted according to:

```
\glsgenentryfmt
```

This is the generic regular entry format. It's encapsulated with \glsxtrregularfont, but note that if the entry is an abbreviation, it will still use the abbreviation style formatting

commands, which are contained within the first, firstplural, text and plural field values.

The generic regular entry format \glsgenentryfmt is provided by the base glossaries package, but is redefined by glossaries-extra to support inner formatting (§5.5.3) and accessibility (§9), if required.

Abbreviations that aren't considered regular, are formatted according to:

```
\glsxtrgenabbrvfmt
```

This is the generic non-regular abbreviation format. It's encapsulated with \glsxtrabbreviationfont. Unlike \glsgenentryfmt this doesn't reference the first, firstplural, text or plural fields, but instead uses the abbreviation formatting commands \glsxtrfullformat, \glsxtrsubsequentfmt and their plural and case-changing variants.

If you want to define your own custom entry format, you will need to consider whether or not your format should support regular and non-regular abbreviation styles. Further detail can be found in the documented code:

```
texdoc glossaries-extra-code
```

5.6. Hyperlinks

The \gls-like and \glstext-like commands will automatically create a hyperlink by default, if hyperref has been loaded (before glossaries/glossaries-extra). The hyperlink can be switched off with hyper=false but will also be switched off if the entry was assigned to an ignored glossary that was defined with the unstarred \newignoredglossary.

> The link text refers to content produced by the \gls-like and \glstext-like commands that has the *potential* to be a hyperlink. That is, if hyperlinks are enabled that content will default to being the hyperlink text. (The post-link text is additional content that may be automatically appended after the link text.) The target is typically the entry's line in the glossary list or to its standalone definition (see §8.5). Link counting (described in §6.2) counts the number of instances of link text not the number of actual hyperlinks.

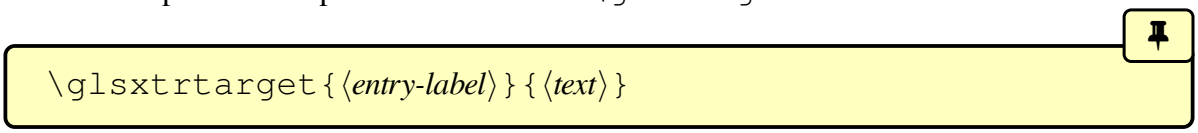
The hyperfirst=false package option and the category attributes nohyper, nohyperfirst and nohypernext can also be used to automatically switch off the hyperlink. See also the hyperoutside option that determines whether the hyperlink should be inside or outside of the outer formatting.

The target for an entry with the label ⟨*entry-label*⟩ is in the form ⟨*prefix*⟩⟨*entry-label*⟩. The ⟨*prefix*⟩ is normally \glolinkprefix but may be changed with the prefix option when

displaying the glossary. The target can also be changed to a link to an external file with the `targeturl` category attribute.

The hyperlink target is usually created by `\glstarget` which is used by all the predefined glossary styles and by the standalone commands, such as `\GlsXtrStandaloneEntryName`. This can result in duplicate targets if you have multiple glossaries or both standalone entries and a glossary. There are ways of getting around this, such as changing the target prefix or using `target=false` when displaying the glossary.

Another option is to replace all instances of `\glstarget` with:



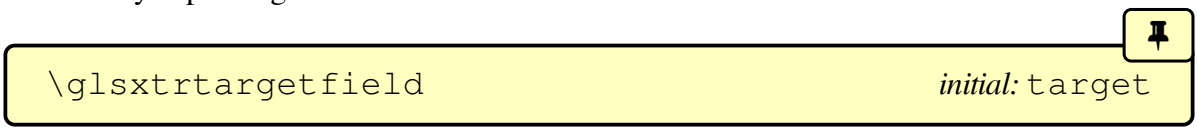
`\glsxtrtarget{`⟨*entry-label*⟩`}{`⟨*text*⟩`}`

This can simply be done by redefining `\glstarget` to use `\glsxtrtarget`. For example:

```
\renewcommand{\glstarget}{\glsxtrtarget}
```

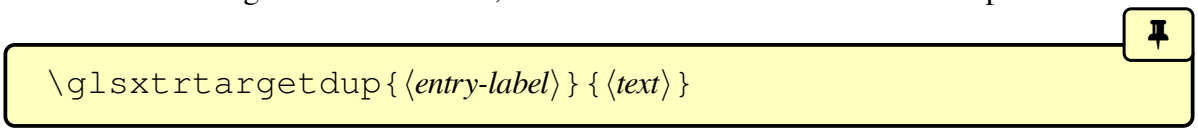
Note that it won't work to have some instances of the original `\glstarget` and then switch to `\glsxtrtarget`, as then it won't have a record of the targets that have already been created.

This new command behaves in a similar manner to `\glstarget` but first tests the field obtained by expanding:



`\glsxtrtargetfield` *initial:* `target`

By default, this expands to `target`. If this field is undefined (according to `\GlsXtrIfFieldUndef`) the target will be created in the way that `\glstarget` would ordinarily create it (if hyperlinks are enabled). The field will then be set to the target. If the field has been defined then the target won't be created, in which case the fallback action is implemented with:



`\glsxtrtargetdup{`⟨*entry-label*⟩`}{`⟨*text*⟩`}`

This simply expands to ⟨*text*⟩ by default. The following will instead create a link back to the actual target:

```
\renewcommand{\glsxtrtargetdup}[2]{\glslink{#1}{#2}}
```

5.7. Label Prefixes

It's possible that you may want to prefix labels to ensure uniqueness. For example, this manual references both the `\makeglossaries` command and the `makeglossaries` Perl script. They are both defined as glossary entries, but they can't both have the label `makeglossaries`. This manual uses `bib2gls` and is quite complicated, but a simplified version is as follows:

```
\newcommand{\csfmt}[1]{\texttt{\#1}}
\newcommand{\appfmt}[1]{\texttt{#1}}
\newglossaryentry{cs.makeglossaries}{
 name={\csfmt{makeglossaries}},
 description={}}
\newglossaryentry{app.makeglossaries}{
 name={\appfmt{makeglossaries}},
 description={}}
```

So the label `cs.makeglossaries` refers to `\makeglossaries` (the command sequence) and the label `app.makeglossaries` refers to `makeglossaries` (the application). If you have a lot of prefixes like this, you may prefer to have a command that automatically adds the prefix. For example,

```
\newcommand*{\cs}[2][]{\gls[#1]{cs.#2}}
```

The problem with this is that the custom command `\cs` doesn't allow for the *, + and ⟨*alt-mod*⟩ modifiers (such as `\gls*` or `\gls+`). Instead you can use:

```
\glsxtrnewgls[⟨default-options⟩]{⟨prefix⟩}{⟨cs⟩}
```

which defines the command

```
⟨cs⟩⟨modifier⟩[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]
```

that behaves like

```
\gls⟨modifier⟩[⟨default options⟩,⟨options⟩]{⟨prefix⟩⟨entry-label⟩}[⟨insert⟩]
```

For example:

```
\glsxtrnewgls{cs.}{\cs}
```

or (to default to no hyperlinks)

```
\glsxtrnewgls[hyper=false]{sym.}{\cs}
```

now you can use `\cs+{M}` to behave like `\gls+{cs.M}`.

If you also want the plural and sentence case versions you can use

```
\glsxtrnewglslike[⟨default-options⟩]{⟨prefix⟩}{⟨\gls-like cs⟩}{⟨\glspl-like cs⟩}{⟨\Gls-like cs⟩}{⟨\Glspl-like cs⟩}
```

For example:

```
\glsxtrnewglslike[hyper=false]{idx.}{\idx}{\idxpl}
{\Idx}{\Idxpl}
```

For the all caps versions:

```
\glsxtrnewGLSlike[⟨default-options⟩]{⟨prefix⟩}{⟨\GLS-like cs⟩}{⟨\GLSpl-like cs⟩}
```

For example:

```
\glsxtrnewGLSlike[hyper=false]{idx.}{\IDX}{\IDXpl}
```

For commands that require the link text to be specified, you can use:

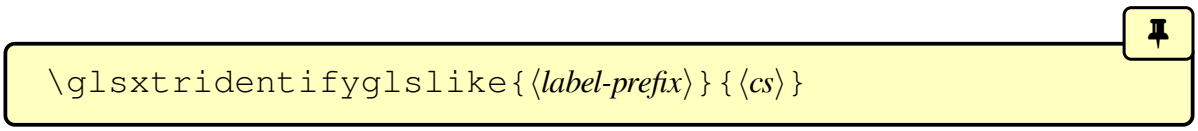
```
\glsxtrnewglslink[⟨default-options⟩]{⟨prefix⟩}{⟨cs⟩}
```

which defines ⟨*cs*⟩ [⟨*options*⟩] {⟨*label*⟩} {⟨*text*⟩} to behave like `\glslink`[⟨*default-options*⟩, ⟨*options*⟩] {⟨*prefix*⟩⟨*label*⟩} {⟨*text*⟩}, or

```
\glsxtrnewglsdisp[⟨default-options⟩]{⟨prefix⟩}{⟨cs⟩}
```

which defines ⟨*cs*⟩ [⟨*options*⟩] {⟨*label*⟩} {⟨*text*⟩} to behave like `\glsdisp`[⟨*default-options*⟩, ⟨*options*⟩] {⟨*prefix*⟩⟨*label*⟩} {⟨*text*⟩}.

If you are using `bib2gls`, it can pick up the custom commands that are defined using the above, so it can detect dependencies when it parses fields such as `description`. If you provide your own custom command with just `\newcommand` that has syntax that starts with [⟨*options*⟩]{⟨*entry-label*⟩}, then you can notify `bib2gls` using:

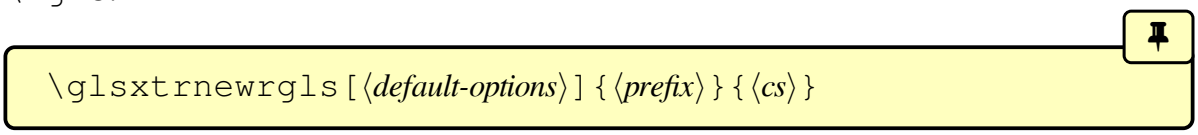


\glsxtridentifyglslike{⟨*label-prefix*⟩}{⟨*cs*⟩}

where ⟨*label-prefix*⟩ is the prefix to apply to the label that's passed to the command ⟨*cs*⟩. The information is written to the aux file so that bib2gls can add the given command to those it looks for when searching for dependencies.

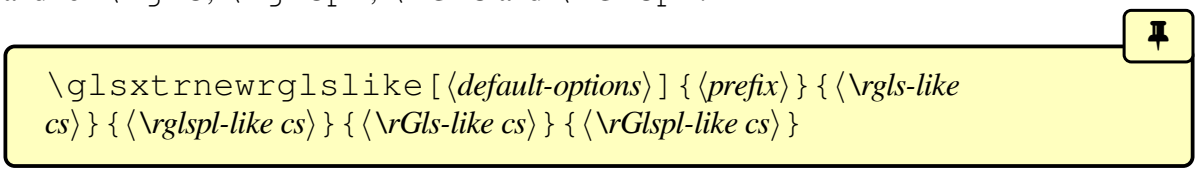
Another possibility when using bib2gls is to set up known label prefixes, see §11.6.7 for further details.

If you use bib2gls with record counting, there are commands to \glsxtrnewgls for \rgls:



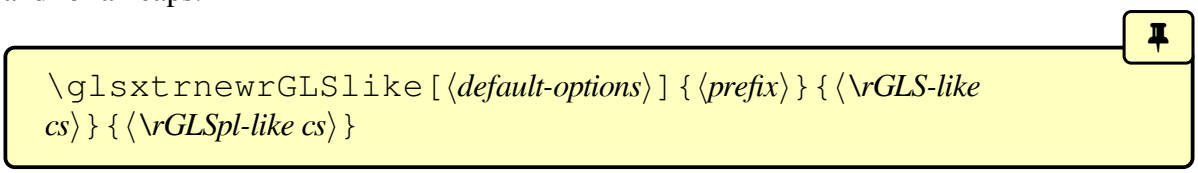
\glsxtrnewrgls[⟨*default-options*⟩]{⟨*prefix*⟩}{⟨*cs*⟩}

and for \rgls, \rglspl, \rGls and \rGlspl:



\glsxtrnewrglslike[⟨*default-options*⟩]{⟨*prefix*⟩}{⟨*\rgls-like cs*⟩}{⟨*\rglspl-like cs*⟩}{⟨*\rGls-like cs*⟩}{⟨*\rGlspl-like cs*⟩}

and for all caps:



\glsxtrnewrGLSlike[⟨*default-options*⟩]{⟨*prefix*⟩}{⟨*\rGLS-like cs*⟩}{⟨*\rGLSpl-like cs*⟩}

Defining commands in this manner (rather than simply using \newcommand) also allows the command to be identified as a sentence case blocker to prevent the label from being converted or, in the case of \glsxtrnewglslike and \glsxtrnewrglslike, as a mapping. See §5.2 for further details.

5.8. Indexing

Indexing is normally performed implicitly by the \gls-like and \glstext-like commands, but this action can be prevented, such as by using the option noindex=true. These commands also generate text (the link text, §5.5). If you want to simply index an entry (to ensure that an entry is shown in the glossary) without producing any text then you can use \glsadd. Indexing is also performed by cross-referencing commands, such as \glssee. In the case of makeindex, \glssee simply behaves like \glsadd with a special format and the location set to Z (which pushes it to the end of the location list). Entries in ignored glossaries can only be indexed with bib2gls.

If you want *all* defined entries to appear in the glossary, regardless of whether or not they have been used in the document, then you can use \glsaddall or \glsaddallunused

(both provided by the base glossaries package). These both iterate over all entries (in all non-ignored glossaries). In the first case (`\glsaddall`), every entry is indexed with the `\glsadd` options provided in the optional argument of `\glsaddall`. In the second case (`\glsaddallunused`), only those entries that haven't been marked as used so far will be indexed using `\glsadd[format=glsignore]{`⟨*label*⟩`}`. See the glossaries manual for further details of those commands.

The glossaries−extra package provides a similar command:

`\glsaddallunindexed[`⟨*glossary types*⟩`]`

This is like `\glsaddallunused` but indexes all entries that haven't been indexed so far (again using the option `format=glsignore`). This is preferable to `\glsaddallunused` if you have to reset the first use flag for any entries. As with `\glsaddallunused`, if this command is required, it should be placed near the end of the document. Indexing any entries after either of these commands are used will cause spurious commas in the location lists.

Iterative commands such as `\glsaddall`, `\glsaddallunused` and `\glsaddallunindexed` should not be used with `bib2gls`. Use the `selection=all` option instead.

If you want to index a specific subset of entries, rather than all entries for a given glossary, you can use:

`\glsaddeach[`⟨*options*⟩`]{`⟨*entry label list*⟩`}`

This does `\glsadd[`⟨*options*⟩`]{`⟨*entry-label*⟩`}` for each entry in the comma-separated ⟨*entry label list*⟩. This command may be used with `bib2gls`, although it may be simpler to adjust the selection criteria or use filtering.

Explicit ranges can be formed by including `(` and `)` at the start of the `format` value. For example:

```
\glsadd[format=(]{example}
...
\glsadd[format=)]{example}
```

(See the glossaries manual for further details.) However, the isolated open and close parentheses can upset syntax highlighting. So the glossaries−extra package provides the following commands, which automatically add `(` and `)`.

```
\glsstartrange[⟨options⟩]{⟨entry label list⟩}
```

This effectively does:

```
\glsaddeach[⟨options⟩,format=(⟨encap⟩]{⟨entry-label list⟩}
```

```
\glsendrange[⟨options⟩]{⟨entry label list⟩}
```

This effectively does:

```
\glsaddeach[⟨options⟩,format=)⟨encap⟩]{⟨entry-label list⟩}
```

The default value of ⟨*encap*⟩ will be the same as the default number format (which can be changed with `\GlsXtrSetDefaultNumberFormat`). If you want a different default for ranges, use:

```
\GlsXtrSetDefaultRangeFormat{⟨encap⟩}
```

This sets the default format for `\glsstartrange` and `\glsendrange`. Note that this format won't be applied if you explicitly create a range with `\glsadd` or `\glsaddeach`.

Alternatively, you can use `format=`⟨*encap*⟩ in ⟨*options*⟩, but remember that this will need to be the same in both `\glsstartrange` and `\glsendrange`. For example:

```
\glsstartrange[format=hyperbf]{example}
...
\glsendrange[format=hyperbf]{example}
```

This is the same as:

```
\GlsXtrSetDefaultRangeFormat{hyperbf}
\glsstartrange{example}
...
\glsendrange{example}
```

which is the same as:

```
\glsadd[format=(hyperbf]{example}
...
\glsadd[format=)hyperbf]{example}
```

The mandatory argument of \glsstartrange and \glsendrange may be a comma-separated list of entry labels. For example:

```
\glsstartrange{duck,goose}
...
\glsendrange{duck,goose}
```

This is essentially the same as:

```
\glsadd[format=(]{duck}%
\glsadd[format=(]{goose}
...
\glsadd[format=)]{duck}%
\glsadd[format=)]{goose}
```

\GlsXtrAutoAddOnFormat[⟨*entry-label*⟩]{⟨*format list*⟩}{⟨*glsadd options*⟩}

This will make the \gls-like and \glstext-like commands automatically use \glsadd[⟨*glsadd options*⟩]{⟨*entry-label*⟩} whenever a \gls-like or \glstext-like command is used when the format matches one of the formats in the comma-separated ⟨*format list*⟩.

The optional argument ⟨*entry-label*⟩ defaults to \glslabel (which will match the ⟨*label*⟩ that was used with the triggering \gls{⟨*label*⟩} etc) and indicates the entry label to use in \glsadd and so needs to be expandable. The ⟨*format list*⟩ is a comma-separated list of format values that will trigger the automated adding. The ⟨*glsadd options*⟩ are the options to pass to \glsadd with format=⟨*format*⟩ prepended to the list.

For example, with:

```
\GlsXtrAutoAddOnFormat{hyperbf}{counter=chapter}
```

then \gls[format=hyperbf]{sample} will be equivalent to:

```
\glsadd[format=hyperbf,counter=chapter]{sample}\gls
[format=hyperbf]{sample}
```

Note that the explicit range markers will prevent a match unless you include them in ⟨*format list*⟩ (in which case, be sure to add both the start and end formats).

Here's another example:

```
\GlsXtrAutoAddOnFormat[dual.\glslabel]{hyperbf}{}
```

In this case `\gls[format=hyperbf]{sample}` will now be equivalent to:

```
\glsadd[format=hyperbf]{dual.sample}\gls[format=
hyperbf]{sample}
```

`\GlsXtrAutoAddOnFormat` is not applied to `\glsadd` as it could cause an infinite loop.

The effects of `\GlsXtrAutoAddOnFormat` can be cleared with:

```
\GlsXtrClearAutoAddOnFormat
```

Both commands perform local redefinitions and so may be scoped.

In the context of glossaries and glossaries−extra, indexing refers to the mechanism used to ensure that an entry is included in its associated glossary. (If you also want to use `\index`, see §12.) This includes any entries that simply cross-reference another entry. The default is to use `makeindex`, which is a general purpose indexing application. Each time an entry is indexed, a line is added to an associated file that contains the indexing information, which includes the sort value, the hierarchical information (if the entry has a parent) and an associated location (the page number, by default). This information is used to sort the entries and collate the locations into a compact location list. The `xindy` package option switches to using `xindy` syntax, but the process is much the same.

Since both `makeindex` and `xindy` are general purpose indexing applications they require an associated location (or a cross-reference) since indexes are typically used to lookup the locations in the document where the term occurs. Although glossaries are similar to indexes they can simply be used to provide brief summaries of each term without any locations. The way that `makeindex` and `xindy` work means that valid locations (that is, locations that conform to `makeindex`/`xindy` syntax) must be supplied even if no location list is required. If an invalid location is used, an error will occur during the `makeindex`/`xindy` step in the build process, even if the location will eventually be ignored when typesetting the glossary.

All location lists can be suppressed with the `nonumberlist` option (which simply discards the location list for each entry), but there are occasions where only some locations need to be suppressed. The main way of hiding a location is to encapsulate the location with a command that does nothing. The `\glsignore` command is used for this purpose (`format=`

glsignore). However, it's important to remember that even though the location isn't shown, it's still present in the location list. This means that you will end up with spurious commas if there's more than one item in the location list.

The "noidx" method similarly writes indexing information, but in this case the information is written to the aux file. Again, empty locations can cause spurious commas in the location lists.

The only method that recognises \glsignore as a special "ignored location" is bib2gls, where this format will trigger the entry's selection but won't add the ignored location to the location list. This avoids the problem of spurious commas caused by invisible locations.

The location corresponds to a counter. The default is the page counter, but may be changed with the counter package option, the ⟨*counter*⟩ optional argument of \newglossary, the counter key when defining an entry, or the counter option when indexing an entry.

Note that bib2gls v3.0+ converts an empty location (which can occur when the location counter is 0 and should be formatted as a Roman numeral) to an ignored location. For example, if you use counter=part but have \gls before the first \part. An empty location will trigger an error with makeindex and xindy.

Since no entries are defined on the first LaTeX run with bib2gls, there's no way of determining the entry's glossary type or of finding if the entry's counter key has been set. This means that if the counter has been assigned to either the entry's glossary or to the entry itself, the location counter can't be implemented until the entry has been defined. A second build is required to ensure that the locations use the correct counter.

The location counter must expand to syntax that's recognised by the indexing application. This is very restrictive with makeindex, which only recognises Western Arabic (\arabic), lowercase Roman numerals (\roman), uppercase Roman numerals (\Roman), lowercase Basic Latin (\alph) and uppercase Basic Latin (\Alph), with optionally a separator (hyphen by default). With xindy, the syntax must be defined (see the glossaries manual for further details).

There's no restriction on the location syntax with bib2gls. The only limitation is that if bib2gls can't determine an associated numeric value according to its location parser, it won't form ranges. This means that with bib2gls, you can set arbitrary text as the location (that's not related to a counter) with thevalue. You can also use thevalue with makeindex and xindy, but only if the value matches the required location syntax.

Both makeindex and xindy order the locations in the location lists. Example 105 demonstrates this:

105

```
\makeglossaries
\newglossaryentry{sample}{name={sample},
 description={an example}}
\begin{document}
\gls[thevalue=Z]{sample} (Z),  \gls[thevalue=4]
```

```
{sample} (4),
\gls[thevalue=xi]{sample} (xi), \gls[thevalue=2]
{sample} (2),
\gls[thevalue=iii]{sample} (iii), \gls[thevalue=A]
{sample} (A).

\printglossaries
\end{document}
```

Example 105: Location list ordering (makeindex)

sample (Z), sample (4), sample (xi), sample (2), sample (iii), sample (A).

Glossary

sample an example iii, xi, 2, 4, A, Z

With makeindex, the location list is grouped into the different number formats (\roman, \arabic and \Alph), with each group ordered numerically. The same result can be produced with xindy by adding the xindy package option to the above example.

With bib2gls, the location list is always in order of indexing. Example 106 adapts Example 105 to use bib2gls: 106

```
\begin{filecontents*}{\jobname.bib}
@entry{sample,name={sample},
 description={an example}}
\end{filecontents*}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources
\begin{document}
\gls[thevalue=Z]{sample} (Z),  \gls[thevalue=4]
{sample} (4),
\gls[thevalue=xi]{sample} (xi), \gls[thevalue=2]
{sample} (2),
\gls[thevalue=iii]{sample} (iii), \gls[thevalue=A]
{sample} (A).
\printunsrtglossaries
\end{document}
```

Example 106: Location list ordering (bib2gls)

sample (Z), sample (4), sample (xi), sample (2), sample (iii), sample (A).

Glossary

sample an example Z, 4, xi, 2, iii, A

These examples are contrived. For most documents, the order of indexing will likely match the desired location list order.

Another important difference between bib2gls and the other indexing methods is the treatment of cross-references identified by the cross-reference keys see, seealso and alias. With bib2gls, the cross-referencing information is picked up when bib2gls parses the bib file and is used to establish dependencies, which ensures that when entries with cross-references are selected, their cross-referenced entries will also be selected.

With the other methods, cross-references are added to an entry's location list by indexing the entry with a special format. The see, seealso and alias keys automatically trigger this indexing unless autoseeindex=false. See §5.9 for further details.

Every time an entry is indexed, the following hook is also used:

```
\glsxtrdowrglossaryhook{⟨entry-label⟩}
```

This does nothing by default. The argument is the entry's label.

The indexing code is encapsulated with:

```
\glsencapwrcontent{⟨code⟩}
```

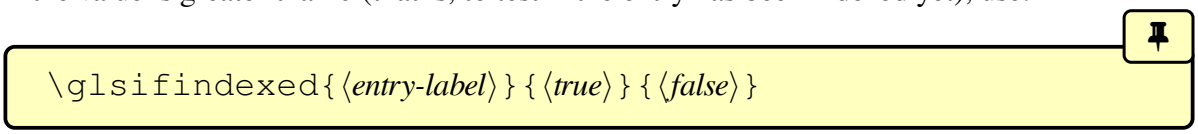
This adds grouping, which helps to prevent spacing issues caused by the whatsit that's created by the indexing.

The base glossaries package always performs the indexing before the link text for the \gls-like and \glstext-like commands. This means that if a page break occurs in the middle of the link text, the location will refer to the page number at the start of the link text (assuming the default page location counter). With glossaries-extra, you can use the option wrgloss=after to have the indexing occur after the link text. The wrgloss attribute can also be used. The default setting is initialised with \glsxtrinitwrgloss (see §5.1.1).

Every time an entry is indexed, an internal field associated with the entry's label is globally updated to keep a count of the number of times the entry has been indexed. The value can be accessed with:

```
\glsentryindexcount{⟨entry-label⟩}
```

This command will expand to 0 if the entry hasn't been indexed or hasn't been defined. To test if the value is greater than 0 (that is, to test if the entry has been indexed yet), use:

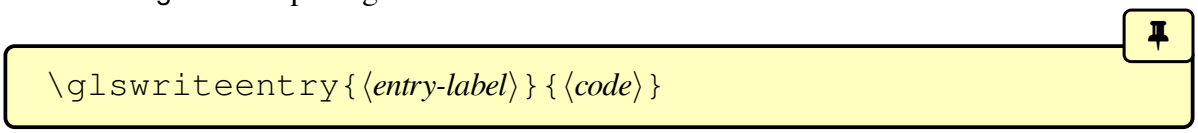


`\glsifindexed{`⟨*entry-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This expands to ⟨*true*⟩ if the entry is defined and has been indexed, otherwise it expands to ⟨*false*⟩. No warning or error occurs if the entry hasn't been defined.

Note that the index count is a running total. This is not the same as the record count saved by `bib2gls`'s `--record-count` switch, which represents the total number of records for the given entry from the previous LaTeX run.

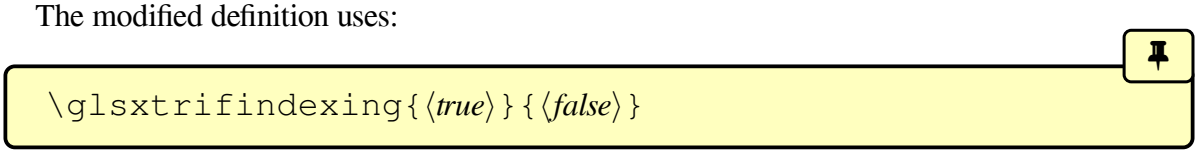
The base glossaries package defines:



`\glswriteentry{`⟨*entry-label*⟩`}{`⟨*code*⟩`}`

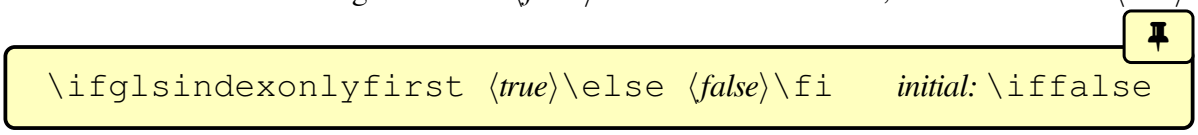
This command conditionally writes the indexing code (supplied by the second argument ⟨*code*⟩). The original definition simply tests whether or not the `indexonlyfirst` setting is on. The glossaries-extra package redefines this command to perform additional checks to determine whether or not the indexing code should be performed.

The modified definition uses:



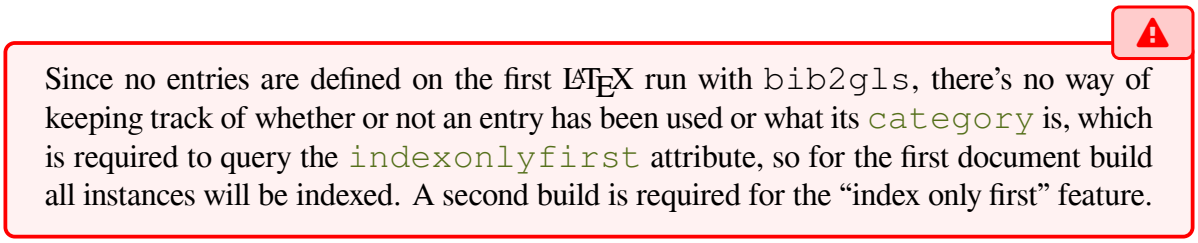
`\glsxtrifindexing{`⟨*true*⟩`}{`⟨*false*⟩`}`

to test the `noindex` setting. This does ⟨*false*⟩ if `noindex=true`, otherwise it does ⟨*true*⟩.



`\ifglsindexonlyfirst` ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`

This is a conditional that corresponds to the `indexonlyfirst` package option. First use is tested using `\GlsXtrIfUnusedOrUndefined` rather than `\ifglsused`. The `indexonlyfirst` attribute is also tested. If the "index only first" setting is on and the entry has been used, ⟨*code*⟩ isn't performed but auto-indexing via `\glsxtrdoautoindex-name` is still performed (see §12).



Since no entries are defined on the first LaTeX run with `bib2gls`, there's no way of keeping track of whether or not an entry has been used or what its `category` is, which is required to query the `indexonlyfirst` attribute, so for the first document build all instances will be indexed. A second build is required for the "index only first" feature.

5.9. Cross-Referencing

The base glossaries package only provides the `see` key, which automatically indexes the cross-reference using `\glssee`. The value of this key isn't saved and can't be accessed later. (The key was simply provided as a shortcut.) The indexing ensures that the cross-reference is shown in the location list.

The auto-indexing feature of the `see` key was intended as a shortcut where only entries required in the document are defined. If you want to have a large file containing entries that may or may not be required in the document, then using `see`, `seealso` or `alias` can cause unwanted entries to appear in the glossary. In this case, see §5.9.1.

The glossaries−extra package saves the value of the `see` key and additionally provides the `seealso` and `alias` keys that perform similar functions. The values of the `see`, `seealso` and `alias` keys can all be accessed at a later point in the document.

If an entry with a cross-reference has been included in the glossary, there's no guarantee that the cross-referenced entry will also be included. It won't be included if it hasn't been indexed anywhere in the document. You can use the `indexcrossrefs` package option to search for cross-references that require indexing at the end of the document, but note that this can be time-consuming if you have a large number of entries.

With `bib2gls` you can simply change the selection criteria (`selection={recorded and deps and see}` or `selection={recorded and deps and see not also}`) to ensure that all cross-referenced entries are included even if they haven't been indexed in the document.

Example (`see` and `seealso` keys):

```
\newglossaryentry{pumpkin}{name={pumpkin},
 description={}}
\newglossaryentry{cucumber}{name={cucumber},
 description={}}
\newglossaryentry{melon}{name={melon},
 description={}}
\newglossaryentry{gourd}{name={gourd},
 description={},
 see={pumpkin,cucumber,melon}}
\newglossaryentry{courgette}{name={courgette},
 description={}}
\newglossaryentry{marrow}{name={marrow},
```

```
 description={},
 seealso={courgette}}
```

When the gourd entry is defined, the cross-reference will automatically be indexed using \glssee. This means that the gourd entry will appear in the glossary, regardless of whether or not it is used in the document, with "*see* pumpkin, cucumber & melon" in the location list. If gourd is also indexed in the document, then those locations will also be added to the gourd's location list.

The cross-referenced entries (pumpkin, cucumber and melon) will only appear in the glossary if they are also indexed in the document. This can be implemented automatically with index-crossrefs.

The seealso key in the marrow entry functions in much the same way, but it is indexed with \glsxtrindexseealso. This means that the marrow entry will have "*see also* courgette" in its location list.

The see key may optionally start with [⟨*tag*⟩] to replace the default \seename tag with ⟨*tag*⟩. The seealso key doesn't permit this. For example, the following is permitted:

```
\newglossaryentry{gourd}{name={gourd},
 description={},
 see={[related topics]pumpkin,cucumber,melon}}
```

but you can't replace see with seealso in the above as it would assume that the first label in the list is [related topics]pumpkin which is incorrect. The tag would have to be removed:

```
\newglossaryentry{gourd}{name={gourd},
 description={},
 seealso={pumpkin,cucumber,melon}}
```

(You could then redefine \seealsoname to related topics, if required or redefine \glsxtruseseealsoformat as applicable.)

Example (alias key):

```
\newglossaryentry{zucchini}{name={zucchini},
 description={},
 alias={courgette}}
```

When the zucchini entry is defined, the alias key will automatically index zucchini with \glssee{zucchini}{courgette}. This means that the zucchini entry will be present in the glossary with "*see* courgette" in the location list. If the zucchini entry is

referenced in the document using a command like \gls, then the hyperlink (if enabled) will go to the courgette entry (not the zucchini entry) but the zucchini entry won't be indexed.

If you want the zucchini entry locations added to the courgette entry, you can redefine \glsxtrsetaliasnoindex (see §5.9.3) or, with bib2gls, use the alias-loc=transfer setting.

With bib2gls, cross-references are selected according to the selection criteria. See the bib2gls manual for further details.

5.9.1. Entries that may not be required

If you have a file containing a large number of entry definitions shared across multiple documents, then the use of the see, seealso or alias key can cause unwanted entries to appear in the document.

Example 107 demonstrates this. Suppose the file myentries.tex contains:

107

```
\newglossaryentry{pumpkin}{name={pumpkin},
 description={}}
\newglossaryentry{cucumber}{name={cucumber},
 description={}}
\newglossaryentry{melon}{name={melon},
 description={}}
\newglossaryentry{gourd}{name={gourd},
 description={},
 see={pumpkin,cucumber,melon}}
\newglossaryentry{cucurbit}{name={cucurbit},
 description={},
 see={gourd}}
\newglossaryentry{courgette}{name={courgette},
 description={}}
\newglossaryentry{marrow}{name={marrow},
 description={},
 seealso={courgette}}
\newglossaryentry{zucchini}{name={zucchini},
 description={},
 alias={courgette}}
\newglossaryentry{broccoli}{name={broccoli},
 description={}}
\newglossaryentry{cauliflower}{name={cauliflower},
```

```
 description={},
 seealso={broccoli}}
```

Some of these entries have a cross-reference key set, but not all of these entries are required in the document:

```
\usepackage[colorlinks]{hyperref}
\usepackage[nostyles,stylemods=bookindex,
 style=bookindex]
 {glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\begin{document}
This document is only discussing \glspl{courgette}
(baby
\glspl{marrow}, also called a \gls{zucchini}),
\glspl{pumpkin} and \glspl{melon}.
\printglossaries
\end{document}
```

Example 107: Cross-references (`autoseeindex=true`)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

cauliflower *see also* broccoli
courgette 1
cucurbit *see* gourd

G

gourd *see* pumpkin, cucumber & melon

M

marrow 1, *see also* courgette
melon 1

P

pumpkin 1

Z

zucchini *see* courgette

Note that the glossary includes cucurbit and gourd, which aren't referenced in the document. They could be useful as a redirect for the reader, but the gourd entry cross-references the cu-

cumber entry, which isn't included in the glossary, so the hyperlink target is undefined. The cauliflower entry has also been included in the glossary, but in this case it's not useful for the reader as neither cauliflower nor broccoli (which it cross-references) are mentioned in the document. As with the cucumber cross-reference, the broccoli cross-reference hyperlink target is undefined.

There are a number of methods to address some of these problems. Example 108 uses the first method, which has the cross-referencing keys in the `tex` file (as above), but disables the auto-indexing: 108

```
\usepackage[colorlinks]{hyperref}
\usepackage[autoseeindex=false,nostyles,
 stylemods=bookindex,
 style=bookindex]{glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\begin{document}
This document is only discussing \glspl{courgette}
(baby
\glspl{marrow}, also called a \gls{zucchini}),
\glspl{pumpkin} and \glspl{melon}.
\printglossaries
\end{document}
```

Example 108: Cross-references (`autoseeindex=false`)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette 1

M

marrow 1

melon 1

P

pumpkin 1

This doesn't show the zucchini entry or any of the cross-references in the glossary because the information hasn't been added to the indexing files.

One way around this is to insert the cross-reference in a post-description hook. Example 109 demonstrates this: 109

```
\usepackage[colorlinks]{hyperref}
\usepackage[autoseeindex=false,nostyles,
  stylemods=bookindex,
  style=bookindex]{glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\glsdefpostdesc{general}{%
 \glsxtrseelists{\glscurrententrylabel}%
}
\begin{document}
This document is only discussing
\glspl{courgette} (baby \glspl{marrow},
also called a \gls{zucchini}),
\glspl{pumpkin} and \glspl{melon}.
\printglossaries
\end{document}
```

Example 109: Cross-references (`autoseeindex=false` and postname hook)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette 1

M

marrow *see also* courgette 1

melon 1

P

pumpkin 1

However, this still doesn't solve the problem that the zucchini entry isn't included in the glossary. It needs to be indexed, but indexing has been suppressed. Firstly, because the automatic indexing triggered by the `alias` key has been suppressed with `autoseeindex=false`, and, secondly, because the presence of the `alias` key automatically suppresses indexing with the `\gls`-like and `\glstext`-like commands. This doesn't cause a problem for the zucchini hyperlink, since the target is courgette (obtained from the `alias` key).

Example 110 demonstrates the second method, which is to not use those keys in the entry definitions and instead use `\glssee` or `\glsxtrindexseealso` within the document. The file `myentries.tex` now contains: 110

```
\newglossaryentry{pumpkin}{name={pumpkin},
 description={}}
\newglossaryentry{cucumber}{name={cucumber},
 description={}}
\newglossaryentry{melon}{name={melon},
 description={}}
\newglossaryentry{gourd}{name={gourd},
 description={}}
\newglossaryentry{cucurbit}{name={cucurbit},
 description={}}
\newglossaryentry{courgette}{name={courgette},
 description={}}
\newglossaryentry{marrow}{name={marrow},
 description={}}
\newglossaryentry{zucchini}{name={zucchini},
 description={}}
\newglossaryentry{broccoli}{name={broccoli},
 description={}}
\newglossaryentry{cauliflower}{name={cauliflower},
 description={}}
```

The document:

```
\usepackage[colorlinks]{hyperref}
\usepackage[nostyles,stylemods=bookindex,
 style=bookindex]
  {glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\glssee{gourd}{pumpkin,melon,courgette}
\glssee{zucchini}{courgette}
\GlsXtrSetField{zucchini}{alias}{courgette}
\begin{document}
This document is only discussing \glspl{courgette}
(baby
\glspl{marrow}, also called a \gls{zucchini}),
\glspl{pumpkin} and \glspl{melon}.
\printglossaries
\end{document}
```

Example 110: Cross-references (no `see`, `seealso` or `alias`)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette 1

G

gourd *see* pumpkin, melon & courgette

M

marrow 1

melon 1

P

pumpkin 1

Z

zucchini *see* courgette

Note that aliases require the `alias` field to be set. In this case, I've set it with `\GlsXtrSetField`. The gourd and zucchini entries have been included in the glossary because they were added with `\glssee`. The other entries are in the glossary because they were indexed when referenced with `\gls` or `\glspl`.

Since cucumber isn't required in the document, I haven't included it in the cross-reference list for gourd. This method is flexible as it allows the cross-referencing to vary between documents. For example, another document may instead have:

```
\glsxtrindexseealso{pumpkin}{courgette,melon}
\glsxtrindexseealso{melon}{pumpkin,courgette}
\glsxtrindexseealso{courgette}{pumpkin,melon}
```

The third method is to switch to `bib2gls`. The file `myentries.tex` can be converted to `myentries.bib` using:

```
convertgls2bib --index-conversion myentries.tex
myentries.bib
```

I've used the option `--index-conversion` (or `-i`) which will use `@index` instead of `@entry` for entries that have an empty description (which is the case in this example). This creates the file `myentries.bib`, which contains the following (space compacted):

```
% Encoding: UTF-8
@index{pumpkin, name={pumpkin}}
@index{cucumber, name={cucumber}}
@index{melon, name={melon}}
@index{gourd, see={pumpkin,cucumber,melon},
 name={gourd}}
@index{cucurbit, see={gourd}, name={cucurbit}}
@index{courgette, name={courgette}}
@index{marrow, name={marrow}, seealso={courgette}}
@index{zucchini, name={zucchini}, alias={courgette}}
@index{broccoli, name={broccoli}}
@index{cauliflower, name={cauliflower},
 seealso={broccoli}}
```

Example 111 adapts the earlier Example 107 to use this new `myentries.bib` file with `bib2gls`: 111

```
\usepackage[colorlinks]{hyperref}
\usepackage[record,nostyles,
 stylemods=bookindex,style=bookindex]
 {glossaries-extra}
\GlsXtrLoadResources[src={myentries}]
\begin{document}
This document is only discussing \glspl{courgette}
(baby
\glspl{marrow}, also called a \gls{zucchini}),
\glspl{pumpkin} and \glspl{melon}.
\printunsrtglossaries
\end{document}
```

In order to support letter groups, `bib2gls` needs to be invoked with the `--group` switch. So if the document file is called `myDoc.tex` then the build process is:

```
pdflatex myDoc
bib2gls --group myDoc
pdflatex myDoc
```

Example 111: Cross-references (bib2gls)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette, 1

M

marrow, 1, *see also* courgette
melon, 1

P

pumpkin, 1

Z

zucchini, *see* courgette

This uses the default `selection={recorded and deps}`, which selects entries that have records, and their dependencies. Records correspond to the usual indexing performed by the `\gls`-like, `\glstext`-like or `\glsadd` commands. With `bib2gls`, the cross-referencing fields don't trigger an index but instead identify dependencies.

Note that the above doesn't include the gourd entry (which cross-references entries that have been indexed but hasn't itself been indexed). The selection criteria can be changed to also include unrecorded entries that cross-reference selected entries. There are two options to choose from: `selection={recorded and deps and see}`, which will apply to all the cross-reference fields (`see`, `seealso` and `alias`), or `selection={recorded and deps and see not also}`, which doesn't consider the `seealso` field.

Example 112 adapts the above Example 111 to change the selection criteria:

112

```
\GlsXtrLoadResources[src={myentries},
 selection={recorded and deps and see}]
```

Example 112: Cross-references (`bib2gls` and `selection=recorded and deps and see`)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette, 1
cucumber
cucurbit, *see* gourd

G

gourd, *see* pumpkin, cucumber & melon

M

marrow, 1, *see also* courgette
melon, 1

P

pumpkin, 1

Z

zucchini, *see* courgette

This now includes the gourd entry because it cross-references pumpkin and melon, which have been recorded in the document. The cucurbit entry is also included because it cross-references the (now selected) gourd entry. Note that the cucumber entry has been selected because the gourd entry depends on it. This means there are no broken links in the glossary, but it looks a bit odd as the cucumber entry has no location list. As from `bib2gls` v3.0, this can be removed with one of the cross-reference pruning options.

Example 113 uses `prune-xr`:

113

```
\GlsXtrLoadResources[
 src={myentries},
 selection={recorded and deps and see},
 prune-xr
]
```

This removes the unnecessary cucumber from the gourd's `see` list, and so cucumber doesn't get selected.

Example 113: Cross-references (`bib2gls` and `selection=recorded and deps and see,prune-xr`)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

C

courgette, 1
cucurbit, *see* gourd

G

gourd, *see* pumpkin & melon

M

marrow, 1, *see also* courgette

melon, 1

P

pumpkin, 1

Z

zucchini, *see* courgette

See the `bib2gls` user manual for further details on the cross-reference selection and pruning options.

5.9.2. Accessing the Cross-Referencing Fields

If you have switched off the indexing of the cross-reference fields (with `autoseeindex=false`) or want to suppress the location lists, then you can adjust the glossary style or hooks to include the cross-references since they won't be shown otherwise.

`\glsxtrseelists{`⟨*entry-label*⟩`}`

If the entry given by ⟨*entry-label*⟩ has the `see`, `seealso` or `alias` fields set, this will display the cross reference according to `\glsxtruseseeformat` (for `see` and `alias`) or `\glsxtruseseealsoformat` (for `seealso`). If any of these fields are set, the list is encapsulated with:

`\glsxtrseelistsencap{`⟨*content*⟩`}`

This simply does a space followed by ⟨*content*⟩. If more than one of the fields are set (not recommended), then they will be displayed in the order: `see`, `seealso` and `alias`. The

entire set will be encapsulated with \glsxtrseelistsencap and each sub-list will be separated with:

\glsxtrseelistsdelim

which defaults to a comma followed by a space.

\glsxtrusesee{⟨*entry-label*⟩}

If the entry given by ⟨*entry-label*⟩ has the see field set, this will display the cross reference according to \glsxtruseseeformat, otherwise this does nothing. An error (or warning with undefaction=warn) will occur if the entry hasn't been defined.

\glsxtrusealias{⟨*entry-label*⟩}

As \glsxtrusesee but for the alias field.

\glsxtruseseealso{⟨*entry-label*⟩}

If the entry given by ⟨*entry-label*⟩ has the seealso field set, this will display the cross reference according to \glsxtruseseealsoformat, otherwise this does nothing. An error (or warning with undefaction=warn) will occur if the entry hasn't been defined.

\glsxtralias{⟨*entry-label*⟩}

This expands to the value of the alias field (which should be a single entry label) or empty if the field isn't set. If the entry isn't defined, this command will expand to \relax (without any error or warning). If you want to first test if the field is set, you can use \glsxtrifhasfield.

\glsxtrseealsolabels{⟨*entry-label*⟩}

This expands to the value of the seealso field (which should be a comma-separated list of entry labels) or empty if the field isn't set. If the entry isn't defined, this command will expand to \relax (without any error or warning). If you want to first test if the field is set, you can use \glsxtrifhasfield.

\glsxtruseseealsoformat{⟨*csv-list*⟩}

This command is used to format a "see also" cross-reference. This is simply defined to do:

```
\glsseeformat[\seealsoname]{⟨csv-list⟩}{}
```

5.9.3. Cross-Reference Indexing

If you are using bib2gls, see the bib2gls user manual for information about the selection, see, seealso, alias, alias-loc options.

The actual indexing of the seealso key is performed with:

```
\glsxtrindexseealso{⟨entry-label⟩}{⟨xr-list⟩}
```

which is analogous to \glssee. As with \glssee, this can also be used explicitly.

With makeindex, \glsxtrindexseealso simply does:

```
\glssee[\seealsoname]{⟨entry-label⟩}{⟨xr-list⟩}
```

With xindy, \glsxtrindexseealso behaves in an analogous way, using the appropriate cross-referencing markup.

```
\glsxtrsetaliasnoindex
```

This hook is used within the \gls-like and \glstext-like commands to automatically switch off the indexing for aliases. (The hook is performed after the options set by \GlsXtrSetDefaultGlsOpts.)

By default, this hook just sets noindex=true. If you would like to add locations to the aliased location list then you can redefine it to use:

```
\glsxtrindexaliased
```

For example:

```
\renewcommand{\glsxtrsetaliasnoindex}{%
 \glsxtrindexaliased}
```

Note that this needs noindex=false to ensure the indexing takes place so don't simply append \glsxtrindexaliased to the definition of \glsxtrsetaliasnoindex.

Don't use the above hooks with bib2gls as this function is disabled with record=only and record=nameref. Use the alias-loc resource option instead.

\glsxtraddallcrossrefs

This is used at the end of the document if indexcrossrefs=true to automatically index any cross-references (identified in the see, seealso and alias fields). This command iterates over all entries in all glossaries and, if an entry has been marked as used, does:

\glsxtraddunusedxrefs

which indexes any labels identified in the cross-reference fields of the entry given by ⟨*entry-label*⟩ that haven't been marked as used.

This can be time consuming if there are a large number of entries defined. If this is the case, you may want to consider switching to bib2gls and use either selection={recorded and deps and see} or selection={recorded and deps and see not also}.

There should be no need to use \glsxtraddallcrossrefs explicitly, but you may want to redefine it to only iterate over specific glossaries. The unused entries are indexed using the glsxtrunusedformat format.

\glsxtrunusedformat{⟨*location*⟩}

This ignores its argument (the location) and just does \unskip.

5.10. First Use Flag

Each entry has an associated first use flag (a conditional or boolean variable), which determines whether or not the entry has been marked as "used". Unsetting this flag means that the entry is marked as used. Resetting the flag means that the entry is marked as unused.

The \gls-like commands (which are the principle method of referencing an entry) all mark the entry as used after the link text is displayed but before the post-link hook is used.

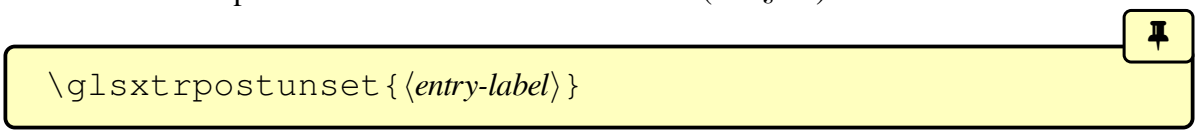
The purpose of this is to allow for additional information that needs to be shown when a term first appears in a document. For example, an abbreviation may need to have its full form shown on the instance. However, there are some cases where that additional information may need to be shown again or where the literal first instance of the term may need to be in its terse form. For example, if the term is used in the front matter.

If any \gls-like commands (which are robust) are used in section headings or captions, they can end up in the table of contents or corresponding "list of …" (such as the list of figures). This

can cause the first use flag to be unset too soon. For these situations, use the commands described in §5.3 instead.

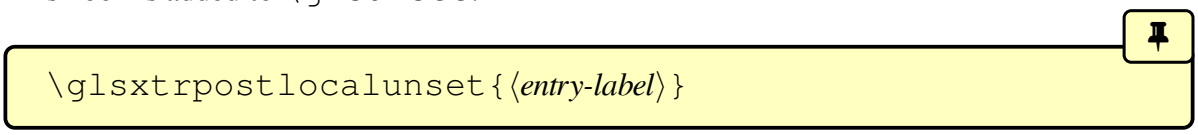
The base glossaries package provides commands to explicitly unset or reset the first use flag either locally (confined to the current scope) or globally. These commands are: `\glsunset` (global unset), `\glslocalunset` (local unset), `\glsreset` (global reset) and `\glslocalreset` (local reset).

The glossaries−extra package adds hooks to the above commands. These do nothing by default, but are modified by `\glsenableentrycount` and `\glsenableentryunitcount` to perform the count increment or reset (see §6.1).



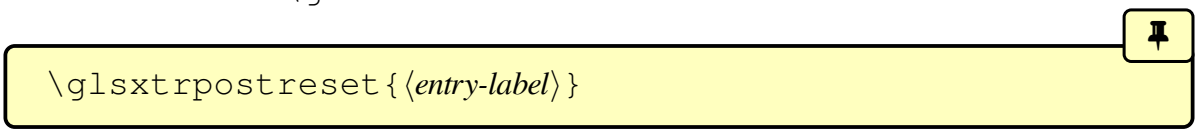
`\glsxtrpostunset{`⟨*entry-label*⟩`}`

This hook is added to `\glsunset`.



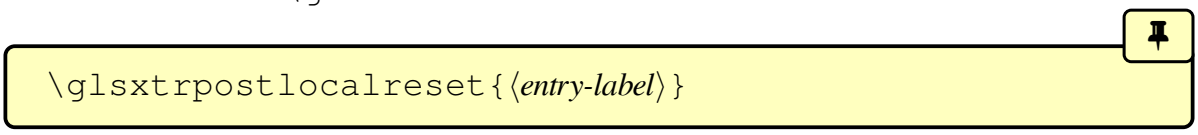
`\glsxtrpostlocalunset{`⟨*entry-label*⟩`}`

This hook is added to `\glslocalunset`.



`\glsxtrpostreset{`⟨*entry-label*⟩`}`

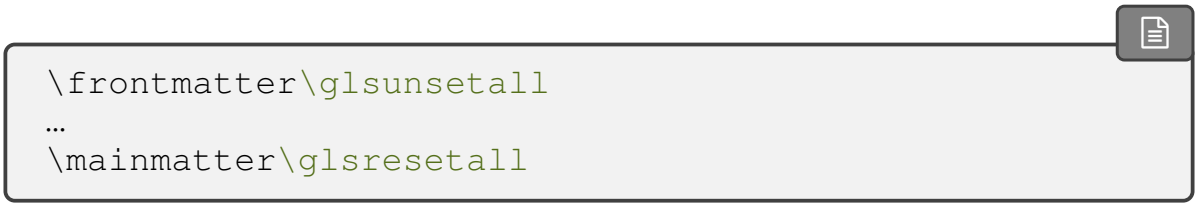
This hook is added to `\glsreset`.



`\glsxtrpostlocalreset{`⟨*entry-label*⟩`}`

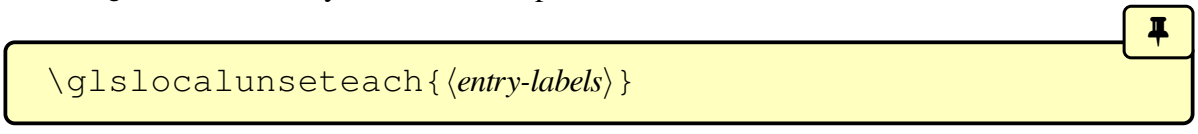
This hook is added to `\glslocalreset`.

The base package also provides commands to unset or reset all entries or all entries within particular glossaries: `\glsunsetall` and `\glsresetall`. For example, if you don't want the first use in the front matter, you can unset all entries at the start of the front matter and reset them at the start of the main matter.



```
\frontmatter\glsunsetall
...
\mainmatter\glsresetall
```

With glossaries−extra you can unset a specific subset of entries.



`\glslocalunseteach{`⟨*entry-labels*⟩`}`

Locally unsets each entry in the given comma-separated list of entry labels.

\glslocalreseteach{⟨*entry-labels*⟩}

Locally resets each entry in the given comma-separated list of entry labels.

You can test if an entry has been marked as used with \ifglsused (but take care if you are using bib2gls or the undefaction=warn option, see below). This command allows the entry display style to vary the link text according to whether or not the entry has been marked as used. However, it can't be used within the post-link hook as by that time, the first use flag will have already been unset.

See the glossaries user manual for further details of the above commands.

Example 114 first uses the "html" entry in the abstract, which shows both the long and the short form, but it would be helpful for the full form to be reshown in the main section about web pages. This is achieved by resetting the first use flag. 114

```
\newabbreviation{html}{HTML}
{hypertext markup language}
\begin{document}
 \begin{abstract}
 This abstract mentions \gls{html}.
 \end{abstract}
 Some casual reference to \gls{html}.
\section{Web Pages}
\glsreset{html}This section is all about \gls{html}.
\end{document}
```

Example 114: Resetting the first use flag (\glsreset)

Abstract

This abstract mentions hypertext markup language (HTML).

Some casual reference to HTML.

1 Web Pages

This section is all about hypertext markup language (HTML).

In the above example, an alternative is to use \glsxtrfull where you particularly want the full form, but some abbreviation styles have a different expansion with the inline \glsxtr-

full form compared with the first use of \gls.

The glossaries-extra package provides the options preunset and prereset, which can be used to unset or reset the first use flag before the link text. This means that in the above example, the line:

```
\glsreset{html}This section is all about \gls{html}.
```

can be replaced with:

```
This section is all about \gls[prereset]{html}.
```

As mentioned above, the first use flag is unset before the post-link hook, so \ifglsused isn't helpful in the post-link hook. Instead, you can use:

```
\glsxtrifwasfirstuse{⟨true⟩}{⟨false⟩}
```

This command is initialised by the \gls-like commands according to the value of the first use flag before the link text. It's also initialised by the \glstext-like commands: not according to the value of the first use flag but according to whether or not the \glstext-like command emulates first use.

For example, \gls will define \glsxtrifwasfirstuse to do its first argument if the first use flag indicates the entry hasn't yet been used, otherwise it will define \glsxtrifwasfirstuse to do its second argument. Whereas \glsfirst will always define \glsxtrifwasfirstuse to do its first argument (unless used with preunset) and \glstext will always define \glsxtrifwasfirstuse to do its second argument (unless used with prereset), regardless of the state of the first use flag.

The preunset and prereset options will additionally redefine \glsxtrifwasfirstuse to match the option. See §5.5.4 for further details about the post-link hook.

If you want to check if the calling command was both the first use and it was a \gls-like command, you can use: \glsxtrifwasglslikeandfirstuse.

The unset function performed by the \gls-like commands before the post-link hook uses the global \glsunset by default. If you want \glslocalunset instead, you can use the local option (provided by the base glossaries package) or postunset=local. To prevent the first use flag from being unset after the link text, use postunset=none.

Example 115 demonstrates locally unsetting the first use flag:

115

```
\newabbreviation{html}{HTML}
{hypertext markup language}
\begin{document}
```

```
{% local scope
\gls[local]{html}. Used? \ifglsused{html}{Yes}{No}.
}% end scope

Used? \ifglsused{html}{Yes}{No}.

\gls[postunset=none]{html}. Used? \ifglsused{html}
{Yes}{No}.
\end{document}
```

Example 115: Local unset

hypertext markup language (HTML). Used? Yes.
Used? No.
hypertext markup language (HTML). Used? No.

If you are using the `undefaction=warn` option (which is automatically implemented by the `record` option), the first use flag is undefined and so is neither true nor false, in which case `\ifglsused` will trigger an error or warning and do neither. In this situation, you may need to use the following command instead.

`\GlsXtrIfUnusedOrUndefined{`⟨*entry-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This does ⟨*true*⟩ if the entry hasn't been defined or hasn't been marked as used, otherwise does ⟨*true*⟩. Note that this command will generate an error or warning (according to `undefaction`) if the entry hasn't been defined, but will still do ⟨*true*⟩. This is more useful than `\ifglsused` with `bib2gls` where the entries are never defined on the first LaTeX run.

5.10.1. Buffering Unsets

Sometimes commands like `\gls` are used in a context where changing a boolean variable can cause things to go wrong. The outer, middle and inner formatting (see §5.5) can be used to change the font for the link text, but it may be that the `\gls`-like command occurs within a block of text that needs to be encapsulated by such a command.

One example of this is using `\gls` in one of the commands provided with the soul package. For example:

```
\ul{Some text about \gls{html}.}
```

This causes the confusing error:

```
Glossary entry `{html}' has not been defined.
```

The simplest workaround is to put `\gls{html}` inside the argument of `\mbox`. For example:

```
\ul{Some text about \mbox{\gls{html}}.}
```

This can work provided it's not the first use of this entry. It if is, then unsetting the first use flag causes a problem and results in the error:

```
! Package soul Error: Reconstruction failed.
```

The glossaries-extra package provides a way of temporarily switching off `\glsunset` so that it just makes a note of the entry's label but doesn't actually perform the change.

`\GlsXtrStartUnsetBuffering` *modifier:* *

This starts the buffering. The unstarred version doesn't check for duplicates, so the internal list may end up with multiple occurrences of the same label. The starred version only adds a label to the internal list if it's not already in it. If you are using entry counting (see §6.1) the unstarred version is preferable to ensure the entry count is correct.

Buffering only applies to the global `\glsunset` and does not affect the local `\glslocalunset`.

The buffer can be locally cleared with:

`\GlsXtrClearUnsetBuffer`

This doesn't stop buffering. It will simply discard the labels that have been buffered so far.

In order to restore the normal behaviour of `\glsunset`, the buffering must be stopped or discarded.

`\GlsXtrStopUnsetBuffering` *modifier:* *

This stops the buffering, restores `\glsunset`, and unsets all the buffered labels. The starred form uses `\glslocalunset` to unset the buffered labels.

Before you stop the unset buffering, you can iterate over the current buffer.

```
\GlsXtrForUnsetBufferedList{⟨handler-cs⟩}
```

This iterates over the currently buffered list of entry labels and performs \⟨*handler-cs*⟩{⟨*entry-label*⟩} for each label, where ⟨*cs*⟩ is a control sequence that takes a single argument. This is best used with the starred version of `\GlsXtrStartUnsetBuffering*` to avoid duplicates.

```
\GlsXtrDiscardUnsetBuffering
```

This discards the buffer and restores `\glsunset` to its normal behaviour.

It's possible to locally unset entries before use (analogous to `preunset=local`) if the entry has already been encountered in the buffer. This will still be problematic for situations where changing a conditional causes a problem, but may be useful in some situations. This feature is enabled with:

```
\GlsXtrUnsetBufferEnableRepeatLocal
```

This may be placed before or after `\GlsXtrStartUnsetBuffering` but the locally collected list of unused entries will be cleared at the start of each instance of `\GlsXtrStartUnsetBuffering`. It will also be cleared by `\GlsXtrClearUnsetBuffer`. All entries that have been marked as unused can be reset with:

```
\GlsXtrResetLocalBuffer
```

This will perform a local reset on all the entries in the "not used" list and do `\GlsXtrClearUnsetBuffer`.

This feature can be switched off with:

```
\GlsXtrUnsetBufferDisableRepeatLocal
```

It's disabled by default.

The way this feature works is as follows (while buffering is active):

1. Each time an entry is referenced with a `\gls`-like command, the `\glsinitreunsets` hook checks if the current entry (identified by `\glslabel`) has been added to the buffer. (Bear in mind that the label is added to the buffer after the link text when `\glsunset` is used.)

   a) If it has been added to the buffer, then this is an indication that the entry has already been used within the buffer zone (that is, an attempt has been made to globally unset the first use flag). A local unset is then performed, which is essentially equivalent to using the `preunset=local` option, so the reference will behave like subsequent use.

b) If it hasn't been added to the buffer, then this is an indication that the entry hasn't already been used within the buffer zone, but it may or may not have been used before the buffering started. If the first use flag indicates that the entry hasn't been used, the entry's label will be added to the "not used" list. The reference will behave like first use, but the unset won't be performed afterwards (because buffering is in progress).

2. The entries that are in the "not used" list can be locally reset and both the buffer and the "not used" list can be cleared with `\GlsXtrClearUnsetBuffer`.

Note that this approach can't be used for situations where the change in conditional causes a problem, but it can be used in situations where the content of an environment or command is repeatedly processed, which upsets the first use flag.

For example, consider the following beamer document:

```
\documentclass{beamer}
\usepackage{glossaries-extra}
\newabbreviation{svm}{SVM}{support vector machine}
\newabbreviation{html}{HTML}
{hypertext markup language}
\begin{document}
\begin{frame}
  \frametitle{Frame 1}
  \begin{itemize}
   \item \gls{html} and \gls{html}
  \end{itemize}
\end{frame}
\begin{frame}
  \frametitle{Frame 2}
  \begin{itemize}
   \item<+-> \gls{svm} and \gls{svm}
   \item<+-> \gls{svm} and \gls{html}
  \end{itemize}
\end{frame}
\frame{\printunsrtglossaries}
\end{document}
```

The first page isn't a problem as the frame doesn't have overlays. The first reference of the "html" entry shows the full form and the next shows just the short form. The second page (which is the first of the overlays of the second frame) correctly shows the full form of the "svm" entry for the first reference and the short form for the second reference, but on the third page (the second of the overlays) now has all instances of "svm" showing as subsequent use (just the short form).

I could put `\glslocalresetall` at the start of the frame, but that would reset the "html" entry as well. Another workaround is to locally reset the first "svm" entry with `pre-`

resetlocal, but that defeats the point of the first use flag, which is intended to keep track of whether or not you have used an entry so that you don't have to.

The following modification places the second frame of the above document inside a buffering zone:

```
\GlsXtrStartUnsetBuffering
\begin{frame}
  \frametitle{Frame 2}
  \begin{itemize}
   \item<+-> \gls{svm} and \gls{svm}
   \item<+-> \gls{svm} and \gls{html}
  \end{itemize}
\end{frame}
\GlsXtrStopUnsetBuffering
```

This ensures that the first use flag isn't reset until after the frame, but it means that all references to the "svm" entry on both the second and third page show the full form.

The "repeat local" function can be used so that repeated references for the same entry can be locally unset before use. This can be enabled with \GlsXtrUnsetBufferEnableRepeatLocal which fixes the second page, but not the third page, which shows all references to "svm" as the short form. What's needed is to locally reset any entries that are in the frame but haven't yet been used ("svm", in this case) at the start of the frame with \GlsXtrResetLocalBuffer. Example 116 does this:

116

```
\GlsXtrStartUnsetBuffering
\GlsXtrUnsetBufferEnableRepeatLocal
\begin{frame}
\GlsXtrResetLocalBuffer
  \frametitle{Frame 2}
  \begin{itemize}
   \item<+-> \gls{svm} and \gls{svm}
   \item<+-> \gls{svm} and \gls{html}
  \end{itemize}
\end{frame}
\GlsXtrStopUnsetBuffering
```

Note that on the first overlay, the buffer and "not used" list are both empty. On the second overlay, the buffer contains the "svm" and "html" labels and the "not used" list just contains the "svm" label. The reset performed by \GlsXtrResetLocalBuffer will reset "svm" and then clear both the buffer and the "not used" list. This means that the first "svm" reference is once again considered first use and it will once again be added to the "not used" list (so that it would be reset again if there was a third overlay).

Example 116: Abbreviations with beamer (unset buffering)

This is quite cumbersome, but these commands could potentially be added to hooks at the start and end of problematic environments (but the buffering needs to be started and ended outside of the repeated content).

Example 117 uses `\mbox` to protect `\gls` within the buffer zone:

117

```
\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{soul}
\usepackage{glossaries-extra}

\newabbreviation{html}{HTML}
{hypertext markup language}

\begin{document}
\GlsXtrStartUnsetBuffering
\ul{Some text about \mbox{\gls{html}}.
Next use: \mbox{\gls{html}}.}
\GlsXtrStopUnsetBuffering

Next use: \gls{html}.
\end{document}
```

Example 117: Buffering first use unsets with `\mbox`

Some text about hypertext markup language (HTML). Next use: hypertext markup languag
Next use: HTML.

Note that the use of `\mbox` prevents line-breaking and the second instance of `\gls{html}` is treated as first use.

Note that since the change in the first use flag now doesn't occur until \GlsXtrStopUnsetBuffering, multiple references of the same term within the buffering zone will always be treated as first use (if the term wasn't used before the buffering started).

Other alternatives include using \protect and inner formatting (see §5.5.3 for limitations) or middle formatting (see §5.5.2) with \GlsXtrExpandedFmt (which can't be used with fragile link text).

Example 118 demonstrates both approaches:

118

```
\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{soul}
\usepackage{glossaries-extra}

\newabbreviation{html}{HTML}
{hypertext markup language}
% custom command to expand content before using \ul:
\newrobustcmd{\xpul}[1]{\GlsXtrExpandedFmt{\ul}{#1}}

\begin{document}
First approach (inner formatting):
{% scope
 \renewcommand{\glsxtrdefaultentrytextfmt}[1]{%
  \ul{#1}}%
 \ul{Some text about \protect\gls{html}.
 Next use: \protect\gls{html}}
}

Next use: \gls{html}.

Second approach (middle formatting with expanded
link text):
\glsresetall
{% scope
 \renewcommand{\glsxtrabbreviationfont}[1]{\xpul{#1}
}%
 \renewcommand{\glsxtrregularfont}[1]{\xpul{#1}}%
 \ul{Some text about \protect\gls{html}.
 Next use: \gls{html}.}
```

```
}

Next use: \gls{html}.
\end{document}
```

Example 118: Alternatives to buffering

First approach (inner formatting): Some text about hypertext markup language (HTML). Next use: HTML
Next use: HTML.
Second approach (middle formatting with expanded link text): Some text about hypertext markup language (HTML). Next use: HTML.
Next use: HTML.

The change in the first use flag isn't the only content within the `\gls`-like commands that can cause a problem. The whatsit caused by indexing can also be problematic. Buffering can also be used to help with that situation. Indexing can be switched off at the start of the buffering and `\GlsXtrForUnsetBufferedList` can be used to perform the indexing outside of the problematic content. Note that this can cause a problem if the location changes (for example, if a page break occurs within the buffering zone).

Buffering can also be used to simply gather the labels that have been referenced with a `\gls`-like command in order to, for example, display a mini-glossary at the end of the block. See for example, Gallery: Mini-Glossary.[1]

5.11. Accessing Fields

See §3.5 for setting fields after an entry has been defined, §5.13 for fields that contain comma-separated lists or whose values may be contained within comma-separated lists, §5.9.2 for cross-referencing fields (`see`, `seealso` and `alias`), and §5.15 for testing field values. See also the base glossaries package's commands, such as `\glsentryname` and `\glsletentryfield`.

`\glsxtrusefield{`⟨*entry-label*⟩`}{`⟨*field-label*⟩`}`

This expands to the value of the field (identified by its internal label ⟨*field-label*⟩) for the entry identified by ⟨*entry-label*⟩. Expands to `\relax` if the field or entry are undefined.

`\Glsxtrusefield{`⟨*entry-label*⟩`}{`⟨*field-label*⟩`}`

[1] dickimaw-books.com/gallery/index.php?label=minigloss

This is like \glsxtrusefield but converts the first character to uppercase using \makefirstuc (provided by mfirstuc) which is robust. If hyperref is loaded, \Glsxtrusefield{⟨entry-label⟩} will use the expandable:

```
\MFUsentencecase{\glsxtrusefield{⟨entry-label⟩}}
```

in a PDF bookmark.

```
\GLSxtrusefield{⟨entry-label⟩}{⟨field-label⟩}
```

This is like \glsxtrusefield but converts the field value to uppercase. See §5.2.3.

```
\glsxtrfieldtitlecase{⟨entry-label⟩}{⟨field-label⟩}
```

This is like \glsxtrusefield but converts the field value to title case. This internally uses:

```
\glsxtrfieldtitlecasecs{⟨content⟩}
```

This converts ⟨content⟩ to title case (expanding the first token once). If \glscapitalisewords has been defined, that will be used, otherwise \capitalisewords will be used.

```
\glsxtrentryparentname{⟨entry-name⟩}
```

Expands to the entry's parent name if defined. Expands to nothing if the entry doesn't have a parent or if the entry isn't defined. If you simply require the parent label then use \glsentryparent or, to first test if the entry has a parent, either use \ifglshasparent or use \glsxtrifhasfield with the field label set to parent.

```
\glsxtrhiername{⟨entry-label⟩}
```

Displays the hierarchical name for the given entry. The cross-reference format \glsseeitemformat may be redefined to use this command to show the hierarchy, if required. Alternatively, you can redefine \glsseeitemformat to \glsxtrseeitemformat which will use \glsxtrhiername if the entry has a parent.

This command has a recursive definition. If the entry given by ⟨entry-label⟩ has a parent, then this command will do \glsxtrhiername{⟨parent-label⟩} for the entry's parent and will then do the separator \glsxtrhiernamesep.

Then, regardless of whether or not the entry has a parent, it will do \glsfmttext{⟨entry-label⟩}, if the entry is an abbreviation (see §1.2.4), or \glsfmtname{⟨entry-label⟩} otherwise.

If hyperref is loaded, \glsxtrhiername will behave as \glsentryname in a PDF bookmark.

\glsxtrhiernamesep

Separator symbol (▹) used between each name in commands like \glsxtrhiername.

\Glsxtrhiername{⟨entry-label⟩}

As \glsxtrhiername but the first name in the list has its first character converted to uppercase using \Glsfmttext or \Glsfmtname (sentence case). If hyperref is loaded, \Glsxtrhiername will expand to:

```
\MFUsentencecase{\glsentryname{⟨entry-label⟩}}
```

in a PDF bookmark. The \makefirstuc mapping from \glsxtrhiername to \Glsxtrhiername is set with \glsmfuaddmap, if supported.

\GlsXtrhiername{⟨entry-label⟩}

As \glsxtrhiername but each name in the list has its first character converted to uppercase using \Glsfmttext or \Glsfmtname.

\GLSxtrhiername{⟨entry-label⟩}

As \glsxtrhiername but the first name in the list is converted to uppercase using \GLSfmttext or \GLSfmtname.

\GLSXTRhiername{⟨entry-label⟩}

As \glsxtrhiername but each name in the list is converted to uppercase using \GLSfmttext or \GLSfmtname (all caps).

5.12. Encapsulation (Formatting) Based on Field Values

These commands assume that a given entry has a special purpose field that's used to store information on how to format text.

5.12.1. Foreign Language Field

\GlsXtrForeignTextField

This command should expand to the internal field label used to store a language tag (such as en-GB or de-CH-1996). The default value is userii (which corresponds to the user2 key).

\GlsXtrForeignText{⟨*entry-label*⟩}{⟨*text*⟩}

If the entry given by ⟨*entry-label*⟩ has the field identified by \GlsXtrForeignTextField set, then this command will encapsulate ⟨*text*⟩ according to the language tag stored in that field.

This uses tracklang's interface to determine the language label that corresponds to the language tag. If the language label can be determined, the ⟨*text*⟩ will be encapsulated with \foreignlanguage otherwise just ⟨*text*⟩ is done.

If \foreignlanguage isn't defined (that is, there's no language support for the document), this command simply does ⟨*text*⟩. If an old version of tracklang is used, this command issues a warning and just does ⟨*text*⟩.

If tracklang can't determine the corresponding language label to use with \foreignlanguage, then a warning is issued with:

\GlsXtrUnknownDialectWarning{⟨*locale*⟩}{⟨*root language*⟩}

where ⟨*locale*⟩ is the language tag supplied in the given field value and ⟨*root language*⟩ is the root language that tracklang has inferred from the tag.

\GlsXtrForeignText requires tracklang v1.3.6+.

Example 119 illustrates this. The language tag is stored in the user2 field and the text in that language is stored in the user1 field. The hook for the long-short-user style's parenthetical material is adjusted to include the non-English text with the applicable hyphenation patterns. 119

```
\usepackage[main=british,brazilian,ngerman]{babel}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-short-user}
\newabbreviation
  [user1={Associação Brasileria de Normas Técnicas},
   user2={pt-BR}
 ]
```

```
 {abnt}{ABNT}
{Brazilian National Standards Organization}

\newabbreviation
 [user1={Deutsches Institut für Normung e.V.},
  user2={de-DE-1996}]
 {din}{DIN}{German Institute for Standardization}

\newabbreviation{tug}{TUG}{\TeX\ User Group}

\renewcommand*{\glsxtruserparen}[2]{%
  \glsxtrfullsep{#2}%
  \glsxtrparen
  {#1%
   \ifglshasfield{\glsxtruserfield}{#2}%
   {, \emph{\GlsXtrForeignText{#2}{%
     \glscurrentfieldvalue}}}%
   {}%
  }%
}

\begin{document}
\gls{abnt}, \gls{din}, \gls{tug}.
\printunsrtglossaries
\end{document}
```

Example 119: Foreign language field encapsulation

Brazilian National Standards Organization (ABNT, *Associação Brasileria de Normas Técnicas*), German Institute for Standardization (DIN, *Deutsches Institut für Normung e.V.*), TEX User Group (TUG).

Glossary

ABNT Brazilian National Standards Organization

DIN German Institute for Standardization

TUG TEX User Group

5.12.2. Associated Entry Format

An entry may have a particular formatting style associated with it (rather than a more general category-wide format). This needs to be provided by a text-block command that takes a single argument. The name (without the leading backslash) should be stored in the field identified by:

`\GlsXtrFmtField` *initial:* `useri`

This command should expand to the internal field label used to store the formatting command's control sequence name. The default value is `useri` (which corresponds to the `user1` key).

`\glsxtrfmt[`⟨*options*⟩`]{`⟨*entry-label*⟩`}{`⟨*text*⟩`}`

This command behaves like:

`\glslink[`⟨*options*⟩`]{`⟨*entry-label*⟩`}{`⟨*fmt-link-text*⟩`}`

where the link text ⟨*fmt-link-text*⟩ is formatted according to:

`\glsxtrfmtdisplay{`⟨*csname*⟩`}{`⟨*text*⟩`}{`⟨*insert*⟩`}`

The default definition simply does `\`⟨*csname*⟩`{`⟨*text*⟩`}`⟨*insert*⟩ where the control sequence name ⟨*csname*⟩ is obtained from the field given by `\GlsXtrFmtField`. If the field hasn't been set, `\@firstofone` is used (which simply does its argument). The unstarred version assumes an empty ⟨*insert*⟩. The default `\glslink` options are given by `\GlsXtrFmtDefaultOptions`.

The post-link hook is suppressed with `\glsxtrfmt`.

If you don't want the complexity of `\glslink`, a partially expandable command is provided that may be used in section headings:

`\glsxtrentryfmt{`⟨*entry-label*⟩`}{`⟨*text*⟩`}`

If hyperref has been loaded, this will expand to:

`\glsxtrpdfentryfmt{`⟨*entry-label*⟩`}{`⟨*text*⟩`}`

within the PDF bookmarks, which just does ⟨*text*⟩. Otherwise `\glsxtrentryfmt` will format ⟨*text*⟩ according to the control sequence name identified in the field given by `\GlsXtrFmtField` (or `@firstofone`, if not set).

Example 120 provides some custom commands with a single argument whose control sequence name is stored in the `user1` field of the associated entry: 120

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\usepackage[colorlinks]{hyperref}
\usepackage[postdot,style=index]{glossaries-extra}
\makeglossaries
\newcommand*{\mtx}[1]{\boldsymbol{#1}}
\newcommand*{\mtxinv}[1]{\mtx{#1}\sp{-1}}

\newglossaryentry{matrix}{%
  name={matrix},
  symbol={\ensuremath{\mtx{M}}},
  plural={matrices},
  user1={mtx},% corresponds to \mtx
  description={rectangular array of values}
}

\newglossaryentry{identitymatrix}{%
  name={identity matrix},
  symbol={\ensuremath{\mtx{I}}},
  plural={identity matrices},
  description={a diagonal matrix with all diagonal
  elements equal to 1 and all other elements
  equal to 0}
}

\newglossaryentry{matrixinv}{%
  name={matrix inverse},
  symbol={\ensuremath{\mtxinv{M}}},
  user1={mtxinv},% corresponds to \mtxinv
  description={a square \gls{matrix} such that
   $\mtx{M}\mtxinv{M}=\glssymbol{identitymatrix}$}
}
\begin{document}
A \gls{matrix} is denoted \glssymbol{matrix}.
The inverse is denoted \glssymbol{matrixinv}.
\[
  \glsxtrfmt{matrix}{A} \glsxtrfmt{matrixinv}{A}
  = \glssymbol{identitymatrix}
```

```
\]
Compare $\glsxtrfmt{matrix}{A}[_0]$
with $\glsxtrfmt*{matrix}{A}[_0]$.
\printglossaries
\end{document}
```

Example 120: Storing a formatting command in a field

A matrix is denoted $\boldsymbol{M}$. The inverse is denoted $\boldsymbol{M}^{-1}$.

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$$

Compare $\boldsymbol{A}[_0]$ with $\boldsymbol{A}_0$.

Glossary

identity matrix ($\boldsymbol{I}$) a diagonal matrix with all diagonal elements equal to 1 and all other elements equal to 0. 1

matrix ($\boldsymbol{M}$) rectangular array of values. 1
matrix inverse ($\boldsymbol{M}^{-1}$) a square matrix such that $\boldsymbol{M}\boldsymbol{M}^{-1} = \boldsymbol{I}$. 1

Note the difference between using `\glsxtrfmt*` (which has a final optional argument) vs `\glsxtrfmt` (which doesn't, so the following square brackets are considered part of the subsequent text).

There are also sentence case versions of the above commands:

`\Glsxtrfmt[⟨options⟩]{⟨entry-label⟩}{⟨text⟩}`

This is simply a shortcut for:

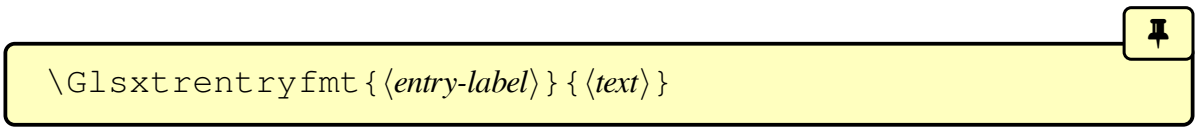
`\glsxtrfmt[⟨options⟩]{⟨entry-label⟩}{\glssentencecase{⟨text⟩}}`

Similarly for the starred form:

`\Glsxtrfmt*[⟨options⟩]{⟨entry-label⟩}{⟨text⟩}[⟨insert⟩]`

which is a shortcut for:

`\glsxtrfmt*[⟨options⟩]{⟨entry-label⟩}{\glssentencecase{⟨text⟩}}[⟨insert⟩]`

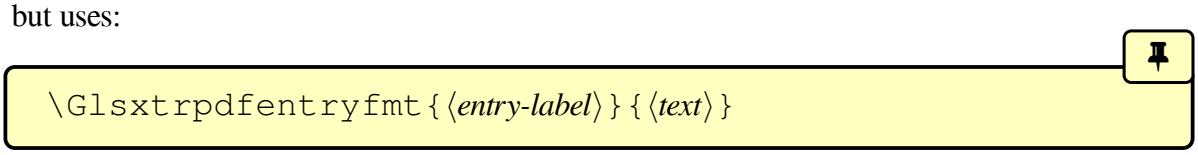


\Glsxtrentryfmt{⟨entry-label⟩}{⟨text⟩}

This is a shortcut for

\glsxtrentryfmt{⟨entry-label⟩}{\glssentencecase{⟨text⟩}}

but uses:



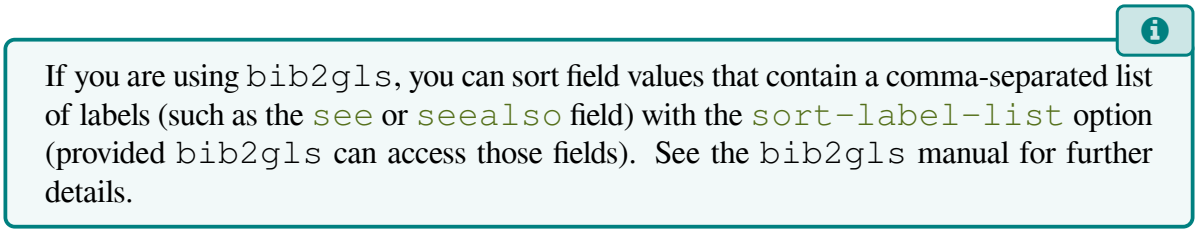
\Glsxtrpdfentryfmt{⟨entry-label⟩}{⟨text⟩}

for the PDF bookmarks. This uses \MFUsentencecase to perform the case-change, which is expandable.

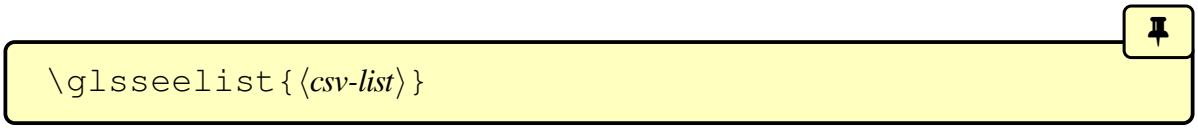
If you are writing \glsxtrfmt or \glsxtrentryfmt explicitly in the document text, you can, of course, enter the appropriate case in ⟨text⟩ directly. The purpose of providing the sentence case commands is to enable a mapping to be setup with \MFUaddmap in the event that \glsxtrfmt or \glsxtrentryfmt occur at the start of content, such as another entry's description, that will have sentence case automatically applied. This will require mfirstuc v2.08+ to support the mapping. See the mfirstuc manual for further details.

5.13. Comma-Separated Lists

These commands are for field values that are comma-separated lists (for example, the field has been constructed with \glsxtrapptocsvfield) or for testing if field values are contained within comma-separated lists.

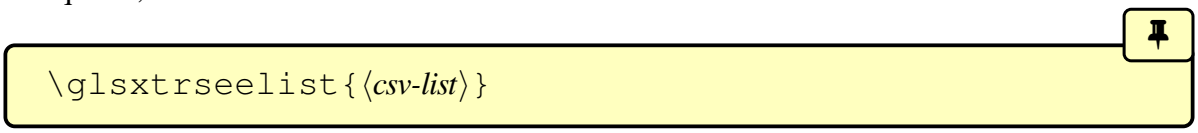


If you are using bib2gls, you can sort field values that contain a comma-separated list of labels (such as the see or seealso field) with the sort-label-list option (provided bib2gls can access those fields). See the bib2gls manual for further details.



\glsseelist{⟨csv-list⟩}

This is provided by the base glossaries package to format the entry labels in see cross-reference list. (It's used internally by \glsseeformat, which adds the *see* prefix.) It may also be used for any comma-separated list of entry labels. Note that the argument isn't expanded. If expansion is required, use:



\glsxtrseelist{⟨csv-list⟩}

This fully expands its argument and passes the result to \glsseelist. With just the base glossaries package, each item is encapsulated with \glsseeitem. The glossaries−extra package redefines \glsseelist to make it more flexible and provides additional commands to further customize the formatting.

\glsxtrtaggedlist{⟨*singular tag*⟩}{⟨*plural tag*⟩}{⟨*label prefix*⟩}{⟨*csv-list*⟩}

This is a similar command that has an initial tag inserted before the start of the list. If the list only contains one element, the ⟨*singular tag*⟩ is used. If the list contains more than one element, the ⟨*plural tag*⟩ is used. The separator between the tag and the list is given by:

\glsxtrtaggedlistsep *initial:* \space

The separators between the elements of the list and the formatting of each list element is as for \glsseelist (see below). If the list is empty, nothing is displayed. The ⟨*label prefix*⟩ is inserted before the current item in the list to form the entry label.

Spaces in ⟨*csv-list*⟩ are significant. Avoid unwanted leading or trailing spaces and empty labels.

\glsseeitemformat{⟨*entry-label*⟩}

The base glossaries package just uses \glsentryname or \glsentrytext in this command. The glossaries−extra package redefines this so that it does:

```
\ifglshasshort{⟨entry-label⟩}{\glsfmttext{⟨entry-label⟩}}
{\glsfmtname{⟨entry-label⟩}}
```

Note that the use of \glsfmttext rather than \glsentrytext allows the abbreviation style to be used.

\glsxtrseeitemformat{⟨*entry-label*⟩}

For convenience, \glsxtrseeitemformat is provided that will use \glsxtrhiername if the entry has a parent, otherwise it will behave as the above definition of \glsseeitemformat. If you want a hierarchical cross-reference, you can either simply redefine

\glsseeitemformat to use \glsxtrhiername or you can redefine \glsseeitemformat to use \glsxtrseeitemformat if you still want \glsfmttext for top-level abbreviations.

With glossaries-extra, the first item in \glsseelist will be encapsulated with:

\glsseefirstitem{⟨*entry-label*⟩}

The default definition is simply \glsseeitem{⟨*entry-label*⟩} but can be redefined, for example to convert the first character to uppercase if sentence case is required.

If the label corresponds to a multi-entry, \mglsseefirstitem will be used instead (see §7.12). Similarly, \mglsseeitem will be used instead of \glsseeitem for a multi-entry label.

\glsseesep *initial: , ␣*

This is used between each entry in the list, except between the final pair.

\glsseelastsep

This is used between the penultimate and final item in the list. The default definition is:

```
\space\andname\space
```

(\andname is provided by glossaries, if not already defined, and simply expands to \& but it may be defined to expand to something else by another package before glossaries is loaded.)

With glossaries-extra, if there are at least three elements in the list, the separator between the final two elements will be given by:

\glsseelastoxfordsep

This just defaults to \glsseelastsep but may be redefined to include a comma, if preferred.

\glsxtrforcsvfield{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*handler cs*⟩} *modifier: **

This iterates over the comma-separated list stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ and performs ⟨*handler cs*⟩{⟨*element*⟩} for each

element of the list. This command uses \glsxtrifhasfield so the complete list can be obtained with \glscurrentfieldvalue. Does nothing if the field hasn't been set or the entry hasn't been defined. The unstarred version adds implicit grouping. The starred version doesn't.

It's possible to prematurely break the loop at the end of the current iteration with:

```
\glsxtrendfor
```

If nested within another command that also uses \@for, use the unstarred version to localise the break. This command is simply set to \@endfortrue, which is provided by the xfor package.

```
\glsxtrfieldformatcsvlist{⟨entry-label⟩}{⟨field-label⟩}
```

This formats the comma-separated list stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ using datatool–base's \DTLformatlist. This command uses \glsxtrifhasfield so the complete list can be obtained with \glscurrentfieldvalue. This adds implicit grouping. There is no starred version.

Example 121 demonstrates the difference between \glsseelist (which specifically requires a list of labels) and \glsxtrfieldformatcsvlist (which formats an arbitrary list): 121

```
\usepackage[colorlinks]{hyperref}
\usepackage[autoseeindex=false]{glossaries-extra}
\newglossaryentry{example}{name={example},
 description={},
 see={another1,another2}}
\newglossaryentry{another1}{name={another one},
 description={}}
\newglossaryentry{another2}{name={another two},
 description={}}
\begin{document}
\glsxtrapptocsvfield{example}{animals}{duck}
\glsxtrapptocsvfield{example}{animals}{albatross}
\glsxtrapptocsvfield{example}{animals}{arara}
Animal list: \glsxtrfieldformatcsvlist{example}
{animals}

See list: \glsxtrifhasfield{see}{example}
 {\glsxtrseelist{\glscurrentfieldvalue}}{not set}.
```

```
\printunsrtglossaries
\end{document}
```

There's no indexing in this document so I've used `autoseeindex=false` to avoid an error. This means there's no cross-reference list in the glossary but, as demonstrated, the "see" list can be reproduced in the document.

Example 121: Formatting lists contained in field values

Animal list: duck, albatross & arara.
See list: another one & another two.

Glossary

example

another one

another two

This first constructs a comma-separated list in a custom internal field with the label `animals`. There's no associated key that can be used in `\newglossaryentry`. In this case, the field could simply be set in one command. For example:

```
\glsxtrdeffield{example}{animals}
{duck,albatross,arara}
```

The main reason for providing `\glsxtrapptocsvfield` is for the benefit of `bib2gls`, as it sometimes has to construct a field value list while it's writing the `glstex` file, but there may be other uses in complex documents that construct field values through some custom function.

`\GlsXtrIfValueInFieldCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}` *modifier: **

This does ⟨*true*⟩ if the comma-separated list stored in the given field (identified by its internal label) contains the given ⟨*value*⟩ (using `\DTLifinlist` provided by datatool–base) or ⟨*false*⟩ if the value isn't in the list or if the field hasn't been set or the entry hasn't been defined. The unstarred version adds implicit grouping. The starred version doesn't.

This command internally uses `\glsxtrifhasfield`, so take care if it's nested. Within ⟨*false*⟩, you can test if `\glscurrentfieldvalue` is empty or undefined. If it's defined but not empty, then the field has been set but doesn't contain ⟨*value*⟩.

\GlsXtrIfFieldValueInCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨csv-list⟩}{⟨true⟩}{⟨false⟩} *modifier: **

This command is essentially the other way around to the above. In this case, the comma-separated list is provided in the argument ⟨*csv-list*⟩ and the search value is the field's value. This does ⟨*true*⟩ if the value is found in ⟨*csv-list*⟩ or ⟨*false*⟩ if the value isn't in ⟨*csv-list*⟩ or the field isn't set or the entry hasn't been defined. The unstarred version adds implicit grouping. The starred version doesn't. Again, this command internally uses \glsxtrifhasfield, so you can test \glscurrentfieldvalue in ⟨*false*⟩ to determine whether or not the field has been set.

\xGlsXtrIfValueInFieldCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩} *modifier: **

As \GlsXtrIfValueInFieldCsvList but fully expands ⟨*value*⟩ first.

5.14. List Fields

Comma-separated list fields are covered in §5.13. The commands in this section are for fields that store etoolbox internal lists. Elements can be appended to these fields using commands \glsxtrfieldlistadd, described in §3.5. The commands listed below provide an easy interface to iterate over the field values. See the etoolbox documentation for further details about internal lists.

\glsxtrfieldformatlist{⟨entry-label⟩}{⟨field-label⟩}

Formats the list using the same separators as used by datatool's \DTLformatlist. This internally uses etoolbox's \forlistcsloop with the same handler macro as used with \DTLformatlist.

\glsxtrfielddolistloop{⟨entry-label⟩}{⟨field⟩}

This uses etoolbox's \dolistcsloop, which uses the command \do as the handler.

\glsxtrfieldforlistloop{⟨entry-label⟩}{⟨field⟩}{⟨handler-cs⟩}

This uses etoolbox's \forlistcsloop, which uses the ⟨*handler-cs*⟩ as the handler.

\glsxtrfieldifinlist{⟨entry-label⟩}{⟨field⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}

This uses etoolbox's \ifinlistcs to test if ⟨*item*⟩ is in the list.

\glsxtrfieldxifinlist{⟨*entry-label*⟩}{⟨*field*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

This uses etoolbox's \xifinlistcs to test if ⟨*item*⟩ is in the list.

5.15. Field Conditionals

\GlsXtrIfFieldUndef{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if the given field (identified by its internal label) is undefined for the entry given by ⟨*entry-label*⟩. Does ⟨*true*⟩ if the entry doesn't exists or if entry exists but the field hasn't been set. Does ⟨*false*⟩ if the field has been set, even if it has been set to empty. Unlike \glsxtrifhasfield there is no grouping or starred version and no assignment of \glscurrentfieldvalue. This is simply a shortcut that internally uses etoolbox's \ifcsundef. The base glossaries package provides a similar command \ifglsfieldvoid, which uses etoolbox's \ifcsvoid instead.

\glsxtrifhasfield{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
*modifier: **

This tests if the entry given by ⟨*entry-label*⟩ has the field identified by its internal label ⟨*field-label*⟩ set. This is like \ifglshasfield but doesn't produce a warning if the entry or field doesn't exist.

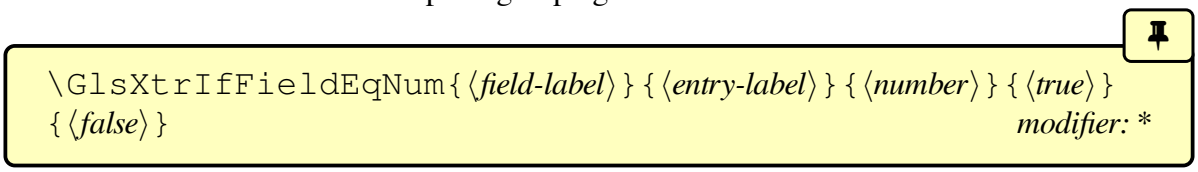
This command first assigns \glscurrentfieldvalue to the field value. If this is defined and not empty, ⟨*true*⟩ is done otherwise ⟨*false*⟩ is done. You can test \glscurrentfieldvalue within ⟨*false*⟩ to find out whether it's undefined or empty using etoolbox's commands, such as \ifundef or \ifdefempty.

The unstarred version adds implicit grouping to make nesting easier. The starred version doesn't (to make assignments easier).

If you are simply displaying the value of the field (for example, in the post-description hook) then use the unstarred version. If you are making an assignment based on the value of the field, then use the starred version.

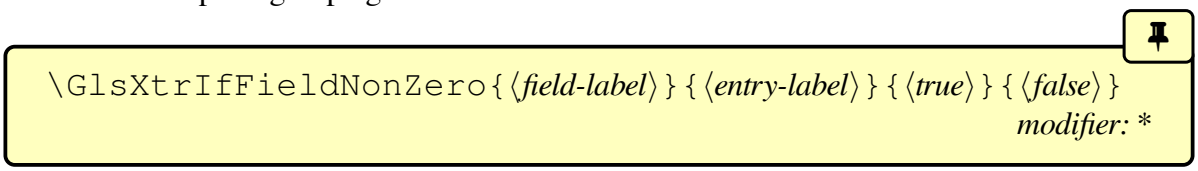
\GlsXtrIfFieldCmpNum{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*op*⟩}{⟨*number*⟩}{⟨*true*⟩}{⟨*false*⟩}
*modifier: **

This command should only be used with fields that contain integer values. It internally uses `\glsxtrifhasfield` (the starred or unstarred version, to match the starred or unstarred version of `\GlsXtrIfFieldEqStr`) and tests if `\glscurrentfieldvalue` is equal to (⟨*op*⟩ is =), less than (⟨*op*⟩ is <) or greater than (⟨*op*⟩ is >) the given number ⟨*number*⟩. If the field is empty or undefined, `\glscurrentfieldvalue` will be set to `0`. Remember that the unstarred version adds implicit grouping.



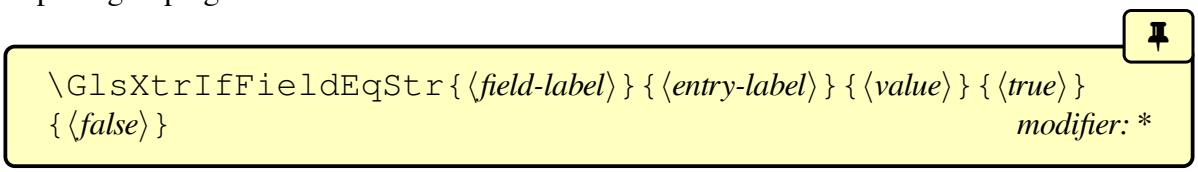
`\GlsXtrIfFieldEqNum{`⟨*field-label*⟩`}{`⟨*entry-label*⟩`}{`⟨*number*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` *modifier: **

This is a shortcut that uses `\GlsXtrIfFieldCmpNum` with ⟨*op*⟩ set to =. The unstarred version adds implicit grouping.



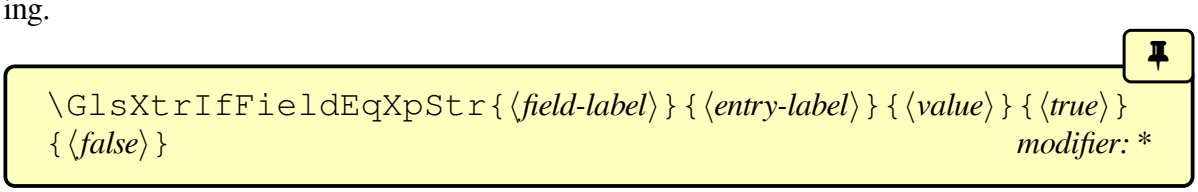
`\GlsXtrIfFieldNonZero{`⟨*field-label*⟩`}{`⟨*entry-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` *modifier: **

This is a shortcut that uses `\GlsXtrIfFieldCmpNum` with ⟨*op*⟩ set to = and the final two arguments swapped. (So it's true if the field value is not zero.) The unstarred version adds implicit grouping.



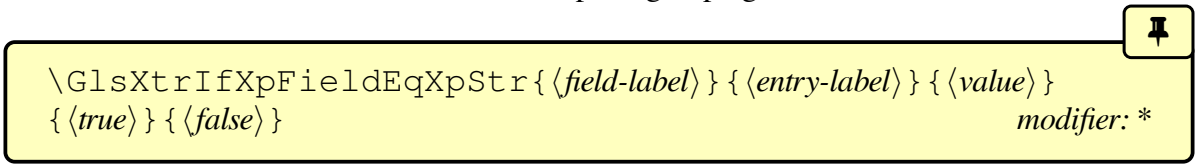
`\GlsXtrIfFieldEqStr{`⟨*field-label*⟩`}{`⟨*entry-label*⟩`}{`⟨*value*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` *modifier: **

This internally uses `\glsxtrifhasfield` (the starred or unstarred version, to match the starred or unstarred version of `\GlsXtrIfFieldEqStr`) and tests if `\glscurrentfieldvalue` is equal to ⟨*value*⟩. Remember that the unstarred version adds implicit grouping.



`\GlsXtrIfFieldEqXpStr{`⟨*field-label*⟩`}{`⟨*entry-label*⟩`}{`⟨*value*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` *modifier: **

This is like `\GlsXtrIfFieldEqStr` but expands the string before the comparison. This also has an starred version that doesn't add implicit grouping.



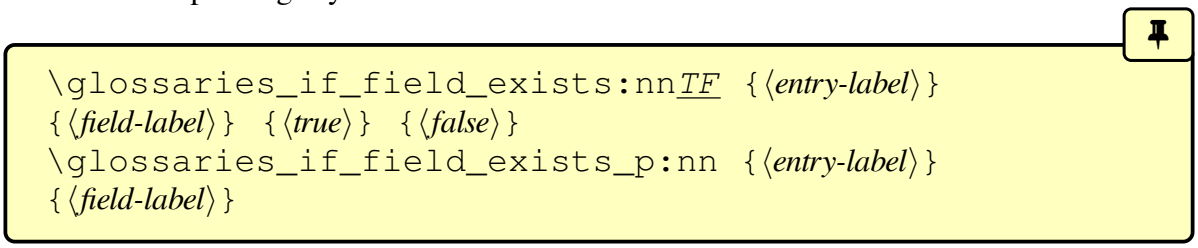
`\GlsXtrIfXpFieldEqXpStr{`⟨*field-label*⟩`}{`⟨*entry-label*⟩`}{`⟨*value*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` *modifier: **

This is like `\GlsXtrIfFieldEqStr` but expands both the field value and the string before the comparison. This also has an starred version that doesn't add implicit grouping.

There are additional conditionals described in the next section.

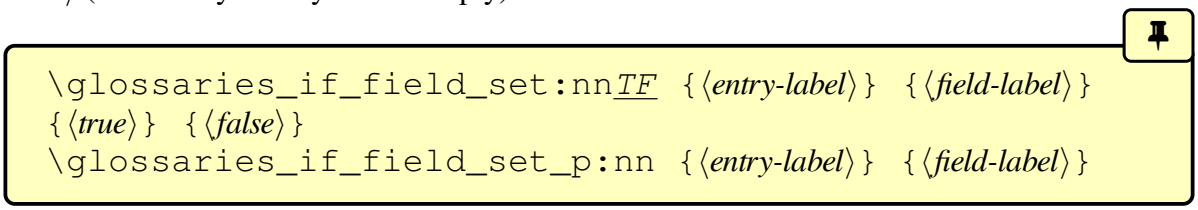
5.16. LaTeX3 Commands

These commands all require LaTeX3 syntax to be on. Unlike most of the commands in the previous section, there is no grouping or starred version and no assignment of `\glscurrentfieldvalue`. In all cases, the field is identified by its internal label. The test for existence means testing if the field exists for the given entry. Existence doesn't automatically mean that there's a corresponding key.



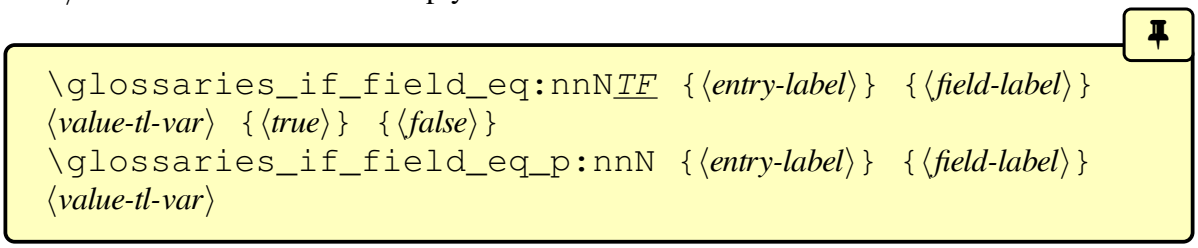
`\glossaries_if_field_exists:nn`*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} {⟨*true*⟩} {⟨*false*⟩}
`\glossaries_if_field_exists_p:nn` {⟨*entry-label*⟩} {⟨*field-label*⟩}

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ (which may or may not be empty).



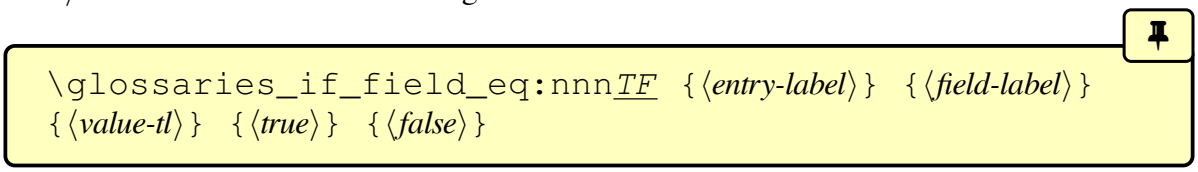
`\glossaries_if_field_set:nn`*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} {⟨*true*⟩} {⟨*false*⟩}
`\glossaries_if_field_set_p:nn` {⟨*entry-label*⟩} {⟨*field-label*⟩}

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ set to a value that is not empty and not `\relax`.



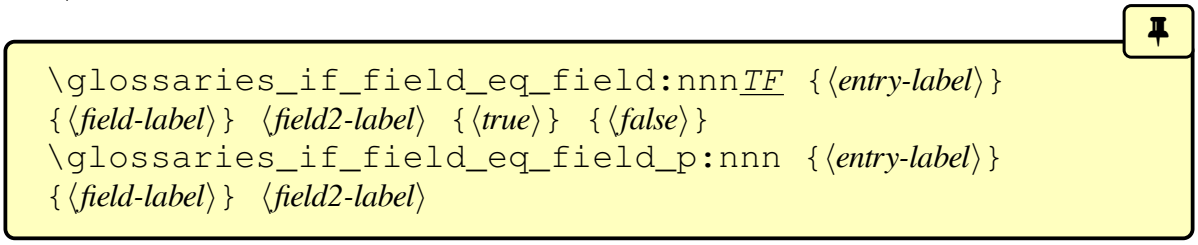
`\glossaries_if_field_eq:nnN`*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*value-tl-var*⟩ {⟨*true*⟩} {⟨*false*⟩}
`\glossaries_if_field_eq_p:nnN` {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*value-tl-var*⟩

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the same value as the given token list variable.



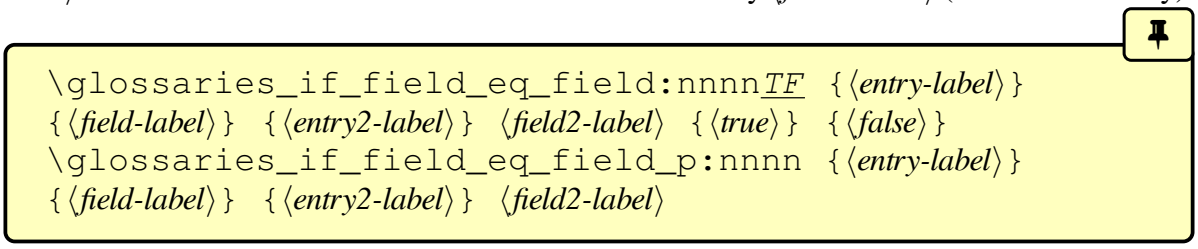
`\glossaries_if_field_eq:nnn`*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} {⟨*value-tl*⟩} {⟨*true*⟩} {⟨*false*⟩}

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the given value.



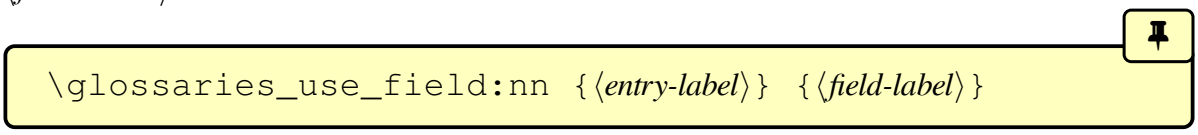
`\glossaries_if_field_eq_field:nnn`*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*field2-label*⟩ {⟨*true*⟩} {⟨*false*⟩}
`\glossaries_if_field_eq_field_p:nnn` {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*field2-label*⟩

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the same value as the internal field identified by ⟨*field2-label*⟩ (for the same entry).



```
\glossaries_if_field_eq_field:nnnnTF {⟨entry-label⟩}
{⟨field-label⟩} {⟨entry2-label⟩} ⟨field2-label⟩ {⟨true⟩} {⟨false⟩}
\glossaries_if_field_eq_field_p:nnnn {⟨entry-label⟩}
{⟨field-label⟩} {⟨entry2-label⟩} ⟨field2-label⟩
```

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ and the entry identified by ⟨*entry2-label*⟩ exists and has an internal field identified by ⟨*field2-label*⟩ and both field values are the same.



```
\glossaries_use_field:nn {⟨entry-label⟩} {⟨field-label⟩}
```

Expands to the value of the field identified by its internal field label ⟨*field-label*⟩ for the entry identifed by ⟨*entry-label*⟩. An error will occur if either the field or entry are undefined).

6. Counting References

There are three basic ways of counting entry references:

1. Counting the total number of times `\glsunset` is used (`\glsreset` resets the count unless `\glsresetcurrcountfalse` and is best avoided). This is provided by the base glossaries package and is intended for documents where the term should be displayed differently if it's only been used a certain number of times. The information has to be written to the `aux` file so that it's available on the next LaTeX run.

   This method is extended by glossaries-extra and is described in §6.1. This method relies on the document only using the `\gls`-like commands and is inappropriate with `bib2gls`.

2. Counting the total number of records. This method is only available with `bib2gls` and is intended for documents where the term should be displayed differently if it's only been recorded (indexed) a certain number of times. This is a more efficient method than entry counting. See §11.5 for further details.

3. Counting the number of times the `\gls`-like or `\glstext`-like commands are used. Unlike the other two methods, this just provides a running total rather than the total from the previous LaTeX run. This method is intended to make it more convenient to work with hooks like `\glslinkcheckfirsthyperhook`, `\glslinkpostsetkeys` or `\glslinkpresetkeys`. See §6.2 for further details.

6.1. Entry Counting (First Use Flag)

If you are using `bib2gls`, you need to use record counting instead (see §11.5).

Entry counting is provided by the base glossaries package and is enabled with `\glsenableentrycount`. This keeps a count of the number of times an entry is marked as used, which is done by hooking into the unset and reset commands (see §5.10). The current running total can be obtained with `\glsentrycurrcount`. The total from the end of the previous LaTeX run can be obtained with `\glsentryprevcount`.

Since entry counting relies on the first use flag, it doesn't take the `\glstext`-like commands into account.

Entry counting is incompatible with docdef=true.

The glossaries-extra package modifies \glsenableentrycount to allow for the entrycount attribute. This means that you not only need to enable entry counting with \glsenableentrycount, but you also need to set the entrycount attribute (see below).

Prior to v1.49, the associated counter was reset back to 0 when the first use flag is reset. This behaviour is now only implemented if the following conditional is true:

\ifglsresetcurrcount ⟨*true*⟩\else ⟨*false*⟩\fi *initial:* \iffalse

To (locally) change this conditional to true use:

```
\glsresetcurrcounttrue
```

To (locally) change this conditional to false use:

```
\glsresetcurrcountfalse
```

As from v1.49, the default is now false. Note that this conditional is also available with glossaries v4.50+.

Remember that entry counting only counts the number of times an entry is used by commands that change the first use flag. (That is, all those commands that mark the entry as having been used.) There are many commands that don't modify this flag and they won't contribute to the entry use count.

With just the base glossaries package, the associated entry counting commands, such as \cgls, are only available when entry counting has been activated with \glsenableentrycount. Whereas with glossaries-extra, those commands are always available but behave in the same way as the corresponding \gls-like commands if entry counting hasn't been activated. The commands provided by the shortcuts options, such as \ac are defined to use \cgls instead of \gls etc so you can use them either with or without entry counting.

In order to activate entry counting with glossaries-extra, you not only need to use \glsenableentrycount but also need to specify the trigger value.

\GlsXtrEnableEntryCounting{⟨*category-list*⟩}{⟨*trigger-value*⟩}

This command is provided as a shortcut to activate entry counting and assign the trigger value. This command performs the following:

- enables entry counting with \glsenableentrycount;
- redefines the \gls-like commands to do the equivalent \cgls commands (so you don't need to keep track of which entries have entry counting enabled);
- sets the entrycount attribute to ⟨*trigger-value*⟩ for all the supplied categories;
- disables the unit counting command (which is incompatible).

If you want to have different trigger values for different categories, you can set the entrycount attribute afterwards for the other category. For example:

```
\GlsXtrEnableEntryCounting{abbreviation,acronym}{1}
\glssetcategoryattribute{general}{2}
```

If you use \GlsXtrEnableEntryCounting multiple times, the repeated instances will simply set the entrycount attribute for the listed categories. So the above can also be written as:

```
\GlsXtrEnableEntryCounting{abbreviation,acronym}{1}
\GlsXtrEnableEntryCounting{general}{2}
```

The commands like \cgls behave like the corresponding \gls-like command if the entry count at the end of the previous run was more than a trigger value. With just the base glossaries package, this trigger value is 1. With glossaries−extra you can specify a different value.

The appropriate trigger value must be set for the required category or categories.

As with the \gls-like commands, the \cgls set of commands may also be used with the star (*) or plus (+) modifiers or the modifier given by \GlsXtrSetAltModifier.

If the entry count at the end of the previous run doesn't exceed the trigger value, the corresponding formatting command is used instead. For example, \cgls will use \cglsformat. The complete set of commands are:

\cgls[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like \gls otherwise it uses:

\cglsformat{⟨*entry-label*⟩}{⟨*insert*⟩}

This is redefined by glossaries-extra to test whether or not the entry has the `regular` attribute set or is an abbreviation:

```
\renewcommand*{\cglsformat}[2]{%
  \glsifregular{#1}{\glsentryfirst{#1}}%
  {% not regular
    \ifglshaslong{#1}
      {\glsentrylong{#1}}% has long
      {\glsentryfirst{#1}}% no long value
  }#2%
}
```

This show the first use value if the entry is regular otherwise it will show the long form. The insert is appended at the end.

`\cglspl[`⟨*options*⟩`]{`⟨*entry-label*⟩`}[`⟨*insert*⟩`]` *modifiers:* * + ⟨*alt-mod*⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like `\glspl` otherwise it uses:

`\cglsplformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}`

This is like `\cglsformat` but uses the plural commands.

`\cGls[`⟨*options*⟩`]{`⟨*entry-label*⟩`}[`⟨*insert*⟩`]` *modifiers:* * + ⟨*alt-mod*⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like `\Gls` otherwise it uses:

`\cGlsformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}`

This is like `\cglsformat` but uses the sentence case commands.

`\cGlspl[`⟨*options*⟩`]{`⟨*entry-label*⟩`}[`⟨*insert*⟩`]` *modifiers:* * + ⟨*alt-mod*⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like `\Glspl` otherwise it uses:

`\cGlsplformat{`⟨*entry-label*⟩`}{`⟨*insert*⟩`}`

This is like \cglsformat but uses the plural sentence case commands.

The glossaries-extra package provides some additional commands:

\cGLS[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like \GLS otherwise it uses:

\cGLSformat{⟨entry-label⟩}{⟨insert⟩}

This simply uses \cglsformat converted to uppercase.

\cGLSpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry's category and is exceeded, this behaves like \GLSpl otherwise it uses:

\cGLSplformat{⟨entry-label⟩}{⟨insert⟩}

This simply uses \cglsplformat converted to uppercase.

The test to determine whether or not an entry trips the trigger value is performed by:

\glsxtrifcounttrigger{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}

This obtains the trigger value from the entry's entrycount attribute.

Since these commands require information from the previous LaTeX run, and extra LaTeX call must be added to the build process (before the relevant indexing application).

Example 122 has the trigger value is set to 1. The CSS entry is only used once (which doesn't exceed the trigger). The HTML entry is used twice (which does exceed the trigger). The sample entry is only used once, but entry counting hasn't been enabled on its category (the default general). 122

```
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\GlsXtrEnableEntryCounting{abbreviation}{1}
\newabbreviation{css}{CSS}{cascading style sheet}
```

```
\newabbreviation{html}{HTML}
{hypertext markup language}
\newglossaryentry{sample}{name={sample},
 description={an example}}
\begin{document}
First use: \gls{css}, \gls{html} and \gls{sample}.
Next use: \gls{html}.
\printglossaries
\end{document}
```

If the document is saved in a file called `myDoc.tex` then the build process is:

```
pdflatex myDoc
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
```

Note the second LaTeX call before `makeglossaries`.

Example 122: Entry counting according to category

First use: cascading style sheet, hypertext markup language (HTML) and sample. Next use: HTML.

Glossary

HTML hypertext markup language 1

sample an example 1

Note that the CSS entry only shows the long form, doesn't appear in the glossary and doesn't have a hyperlink. This is because the total count from the previous LaTeX run doesn't exceed the value (1, in this case) that triggers the normal behaviour of `\gls`. The HTML entry has a total count of 2 from the previous LaTeX run, so it's displayed as normal with the full form on first use and has a hyperlink to its entry in the glossary.

The sample entry is only used once, but it has the default `general` category, which doesn't have the `entrycount` attribute set.

Note that if the build process only had one LaTeX call before running `makeglossaries`, the HTML entry would also not appear in the glossary. This is because on the first LaTeX run, the total from the previous run is 0 (because there's no information in the `aux` file).

The glossaries-extra package also provides the ability to count per sectional unit instead:

\glsenableentryunitcount

It's not possible to enable both document-wide entry counting (\glsenableentry-count) and unit entry counting (\glsenableentryunitcount).

The unit entry counting provides separate totals for each section unit. As above, this uses the entrycount attribute to provide the trigger value but also requires the unitcount attribute, which should be set to the name of the appropriate counter, such as section or chapter.

Due to the asynchronous nature of TEX's output routine, discrepancies will occur in page spanning paragraphs if you use the page counter.

As before, there is a command provided to enable the feature and set the corresponding attributes at the same time:

\GlsXtrEnableEntryUnitCounting{⟨*category-list*⟩}{⟨*trigger-value*⟩}{⟨*counter*⟩}

This command performs the following:

- enables unit entry counting with \glsenableentryunitcount;
- redefines the \gls-like commands to do the equivalent \cgls commands (so you don't need to keep track of which entries have entry counting enabled);
- sets the entrycount attribute to the supplied trigger for all the supplied categories;
- sets the unitcount attribute to the supplied counter for all the supplied categories;
- disables the document-wide counting command (which is incompatible).

If you use \GlsXtrEnableEntryUnitCounting multiple times, the repeated instances will simply set the entrycount and unitcount attributes for the listed categories.

The counter value is used as part of a label, which means that \the⟨*counter-name*⟩ needs to be expandable. Since hyperref also has a similar requirement and provides \the⟨*counter-name*⟩ as an expandable alternative, glossaries-extra will use \the⟨*counter-name*⟩ if it exists otherwise it will use \the⟨*counter-name*⟩.

The commands for accessing the totals, \glsentrycurrcount and \glsentryprevcount have different definitions with unit entry counting and will expand to the total for the current unit. The overall totals can be obtained with additional commands:

\glsentryprevtotalcount{⟨*entry-label*⟩}

This expands to the overall total from the previous LaTeX run.

\glsentryprevmaxcount{⟨*entry-label*⟩}

This expands to the maximum per-unit total from the previous LaTeX run.

Example 123 illustrates unit entry counting:

123

```
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\GlsXtrEnableEntryUnitCounting{abbreviation}{2}
{section}

\makeglossaries

% default category={abbreviation}:
\newabbreviation{html}{HTML}
{hypertext markup language}
\newabbreviation{css}{CSS}{cascading style sheet}

% default category={general}:
\newglossaryentry{sample}{name={sample},
 description={sample}}

\begin{document}
\section{Sample}

Used once: \gls{html}.

Used three times: \gls{css} and \gls{css} and
\gls{css}.

Used once: \gls{sample}.

\section{Another Sample}

Used once: \gls{css}.
```

```
Used twice: \gls{html} and \gls{html}.

\printglossaries
\end{document}
```

As before, if the document is in a file called `myDoc.tex` then the build process is:

```
pdflatex myDoc
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
```

Example 123: Entry unit counting (per section) according to category

1 Sample

Used once: hypertext markup language.
Used three times: cascading style sheet (CSS) and CSS and CSS.
Used once: sample.

2 Another Sample

Used once: cascading style sheet.
Used twice: hypertext markup language and hypertext markup language.

Glossary

CSS cascading style sheet 1

sample an example 1

In this document, the CSS entry is used three times in the first section. This is more than the trigger value of 2, so `\gls{css}` is expanded on first use with the short form used on subsequent use, and the CSS entries in that section are added to the glossary. In the second section, the CSS entry is only used once, which trips the suppression trigger, so in that section, the long form is used and `\gls{css}` doesn't get a line added to the glossary file.

The HTML entry is used a total of three times, but the expansion and indexing suppression trigger is tripped in both sections because the per-unit total (1 for the first section and 2 for the second chapter) is less than or equal to the trigger value.

The sample entry has only been used once, but it doesn't trip the indexing suppression because it's in the general category, which hasn't been listed in \GlsXtrEnableEntryUnitCounting.

The per-unit entry counting can be used for other purposes. Example 124 has the trigger value set to zero, which means the index suppression won't be triggered, but the unit entry count is used to automatically suppress the hyperlink for commands like \gls by modifying the \glslinkcheckfirsthyperhook which is used at the end of the macro the determines whether or not to suppress the hyperlink.

124

```
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\GlsXtrEnableEntryUnitCounting{general}{0}{page}
\newglossaryentry{sample}{name={sample},
 description={an example}}

\renewcommand*{\glslinkcheckfirsthyperhook}{%
  \ifnum\glsentrycurrcount{\glslabel}>0
   \setupglslink{hyper=false}%
  \fi
}

\begin{document}

A \gls{sample} entry.  Next use: \gls{sample}.

\newpage
Next page: \gls{sample}.  Again: \gls{sample}.

\printglossaries
\end{document}
```

This only produces a hyperlink for the first instance of \gls{sample} on each page.

The earlier warning about using the page counter still applies. If the first instance of \gls occurs at the top of the page within a paragraph that started on the previous page, then the count will continue from the previous page.

Example 124: Enabling unit counting to hook into hyperlink setting

A sample entry. Next use: sample. Next page: sample. Again: sample.

Glossary

sample an example 1, 2

6.2. Link Counting

As from version 1.26, an alternative method of entry counting is to count the number of times the `\gls`-like or `\glstext`-like commands are used. (The "link" in this method's name refers to the use of the internal command `\@gls@link` not to `\hyperlink` although `\@gls@link` may use `\hyperlink` when displaying the link text.)

To enable link counting use the preamble-only command:

```
\GlsXtrEnableLinkCounting[⟨parent counter⟩]{⟨categories⟩}
```

where ⟨*categories*⟩ is a list of category labels. The optional argument ⟨*parent counter*⟩ may be used to identify a parent counter (which must already be defined). If present, the associated link counter will be reset when the parent counter is incremented. This command automatically sets the `linkcount` attribute for the given categories. If the optional argument is present, it also sets the `linkcountmaster` attribute.

When enabled, the `\gls`-like and `\glstext`-like commands will increment the associated counter using

```
\glsxtrinclinkcounter{⟨counter⟩}
```

This just does `\stepcounter{`⟨*counter*⟩`}` by default but if you need `\refstepcounter` instead, just redefine this command:

```
\renewcommand*{\glsxtrinclinkcounter}[1]{%
 \refstepcounter{#1}}
```

You can access the internal count register using:

```
\GlsXtrLinkCounterValue{⟨entry-label⟩}
```

where ⟨*label*⟩ is the entry's label. This will expand to 0 if the register hasn't been defined.

It's also possible to access the display value (`\the`⟨*counter*⟩) using

```
\GlsXtrTheLinkCounter{⟨entry-label⟩}
```

(This will expand to 0 if the counter hasn't been defined.)

> In order to conserve resources, the counter is only defined when it first needs to be incremented so terms that have been defined but haven't been used in the document won't have the associated count register allocated.

You can test if the counter has been defined using:

```
\GlsXtrIfLinkCounterDef{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
```

This expands to ⟨*true*⟩ if the link counter associated with the entry identified by ⟨*entry-label*⟩ has been defined, otherwise expands to ⟨*false*⟩.

The counter name can be obtained using

```
\GlsXtrLinkCounterName{⟨entry-label⟩}
```

This simply expands to the counter name associated with the entry given by ⟨*entry-label*⟩ without any check for existence. For example, to change the display command (`\the`⟨*counter*⟩) using etoolbox:

```
\csdef{the\GlsXtrLinkCounterName{duck}}%
 {\Roman{\GlsXtrLinkCounterName{duck}}}
```

This is useful if you just want to change the display for specific entries but isn't convenient if you want to change the display for all entries. Instead, it's simpler to redefine `\GlsXtrTheLinkCounter`. For example:

```
\renewcommand*{\GlsXtrTheLinkCounter}[1]{%
 \GlsXtrIfLinkCounterDef{#1}%
 {\Roman{\GlsXtrLinkCounterName{#1}}}%
 {0}%
}
```

In both cases, the redefinition should be implemented after `\GlsXtrEnableLinkCounting`.

Example 125 uses link counting to disable the hyperlink after the first reference. This redefines \glslinkpresetkeys (which is used by both \gls and \glstext) instead of \glslinkcheckfirsthyperhook (which is used by \gls but not by \glstext). 125

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\makeglossaries

\renewcommand*{\glslinkpresetkeys}{%
 \ifnum\GlsXtrLinkCounterValue{\glslabel}>1
  \setupglslink{hyper=false}%
 \fi
}

\GlsXtrEnableLinkCounting{general}

\newglossaryentry{sample1}{name={sample1},
 description={an example}}
\newglossaryentry{sample2}{name={sample2},
 description={another example}}

\newabbreviation{ex}{ex}{example}

\begin{document}
\section{Sample Section}

\Gls{sample1}, \gls{sample2} and \gls{ex}.
\Glstext{sample1} and \gls{ex} again.

\section{Another Sample Section}

\Gls{sample1}, \gls{sample2} and \gls{ex}.

\printglossaries
\end{document}
```

The use of \glslinkpresetkeys means that the options can override this. For example

```
\gls[hyper=true]{sample1}
```

(or simply \gls+{sample1}) will override the hyper=false setting in \glslinkpresetkeys. If \glslinkpostsetkeys is used instead, the hyper=false setting will override the setting provided in the optional argument.

Example 125: Link counting used to selectively suppress hyperlinks

1 Sample Section

Sample1, sample2 and example (ex). Sample1 and ex again.

2 Another Sample Section

Sample1, sample2 and ex.

Glossary

ex example 1

sample1 an example 1

sample2 another example 1

The abbreviation category doesn't have the linkcount attribute set (since it's not listed in the argument of \GlsXtrEnableLinkCounting). This means that \GlsXtrLinkCounterValue always expands to 0 for the abbreviation (ex), so the inequality test:

```
\ifnum\GlsXtrLinkCounterValue{\glslabel}>1
```

will always be false. This means that the abbreviation won't have hyper=false applied. If the test is changed to

```
\ifnum\GlsXtrLinkCounterValue{\glslabel}=1
\else
 \setupglslink{hyper=false}%
\fi
```

Then the abbreviation will always have `hyper=false` applied.

To reset the counter every section use the optional argument to set the parent counter:

```
\GlsXtrEnableLinkCounting[section]{general}
```

7. Multi (or Compound) Entries

Nested entries (where the entry definition references other entries) are discussed in §5.4. This chapter deals with occasions where a term or phrase may consist of multiple sub-terms that are independently defined. (Examples in §7.1.5 and §7.1.6 provide workarounds for nested entries.)

For example, the names of bacteria, such as *Clostridium botulinum* and *Clostridium perfringens*, are made up of the genus (for example, *Clostridium*) and the species (for example, *botulinum* or *perfringens*). The genus is often abbreviated after first use. For example, *C. botulinum*. However, if the name is defined as a single term consisting of both the genus and species then it's not possible to apply the abbreviation when a different species with the same genus is used. Consider the following document:

```
\documentclass{article}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-only-short-only}
\newabbreviation{cbot}{C. botulinum}
{Clostridium botulinum}
\newabbreviation{cperf}{C. perfringens}
{Clostridium perfringens}
\begin{document}
\gls{cbot}, \gls{cbot}, \gls{cperf}.
\end{document}
```

The result is:

Clostridium botulinum, C. botulinum, Clostridium perfringens.

However, it should more typically be:

Clostridium botulinum, C. botulinum, C. perfringens.

In this case, the genus should actually be a separate definition:

```
\documentclass{article}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-only-short-only}
```

```
\newabbreviation{clostridium}{C.}{Clostridium}
\newglossaryentry{botulinum}{name={botulinum},
 description={}}
\newglossaryentry{perfringens}{name={perfringens},
 description={}}
\begin{document}
\gls{clostridium} \gls{botulinum},
\gls{clostridium} \gls{botulinum},
\gls{clostridium} \gls{perfringens}.
\end{document}
```

However, the above is far more cumbersome than the previous example.

This chapter describes a more compact method for dealing with such a situation. Each term should be defined as normal (as in the above example), and a "multi-entry" label is then defined with the list of labels of the entries that need to be referenced.

`\multiglossaryentry[`⟨*options*⟩`]{`⟨*multi-label*⟩`}[`⟨*main-label*⟩`]{`⟨*entry-label-list*⟩`}`

This defines a multi-entry set with the label ⟨*multi-label*⟩, consisting of the entries whose labels are listed in ⟨*entry-label-list*⟩, where the main entry (which must be present in ⟨*entry-label-list*⟩) is identified by ⟨*main-label*⟩. If ⟨*main-label*⟩ is omitted, it's assumed to be the final label in ⟨*entry-label-list*⟩. The main entry is described in more detail in §7.2.

The entries in ⟨*entry-label-list*⟩ must already be defined (using commands like `\newglossaryentry` or `\newabbreviation`).

The ⟨*options*⟩ are a comma-separated list of options to override the current settings and are described in §7.9.

The earlier example can now be modified to include the following:

```
\multiglossaryentry{cbot}{clostridium,botulinum}
\multiglossaryentry{cperf}{clostridium,perfringens}
```

These commands must come after the `clostridium`, `botulinum` and `perfringens` definitions.

Once defined, a multi-entry set can be referenced in the document using commands like:

`\mgls[`⟨*options*⟩`]{`⟨*multi-label*⟩`}[`⟨*insert*⟩`]` *modifiers:* * + ⟨*alt-mod*⟩

This command essentially does \gls{⟨label⟩} for each item in the ⟨*label list*⟩ (with separators, see §7.4). If the final optional argument ⟨*insert*⟩ is provided, it will be applied to the final (non-skipped) element in the list. So the document body in the above example, can be rewritten as:

```
\mgls{cbot}, \mgls{cbot}, \mgls{cperf}.
```

There are some variants of \mgls listed in §7.11. The available ⟨*options*⟩ are listed in §7.10. They are applied after the \multiglossaryentry options and will override settings for the individual entries.

You can't use ⟨*multi-label*⟩ in commands like \gls as this label represents a set of entry labels not a single entry.

The \multiglossaryentry command will generate an error if the label has already been defined as a multi-entry.

\providemultiglossaryentry[⟨*options*⟩]{⟨*multi-label*⟩}[⟨*main-label*⟩]{⟨*entry-label-list*⟩}

This does nothing if a multi-entry set with the given label has already been defined otherwise it will act like \multiglossaryentry. Notes and associated commands applying to \multiglossaryentry also apply to \providemultiglossaryentry unless otherwise stated.

\multiglossaryentry may be placed anywhere after the entries listed in ⟨*label list*⟩ have been defined. A multi-entry label can't be referenced (with commands like \mgls) before it has been defined.

There is limited support for docdef=true. The multi-entry definition can be picked up from the aux file on the next run to allow cross-references in any glossaries that occur at the start of the document. Any changes made with commands like \mglsSetMain won't be carried over to the next run.

By default \multiglossaryentry will be localised to the current scope. If you want to globally define a multi-entry you need to first switch on global definitions with:

```
\multiglossaryentryglobaltrue
```

To switch back to local definitions use:

```
\multiglossaryentryglobalfalse
```

You can test if this setting is on with:

```
\ifmultiglossaryentryglobal ⟨true⟩\else ⟨false⟩\fi
                                                initial: \iffalse
```

If you want to change the multi-entry options (locally) you can use:

```
\mglsSetOptions{⟨multi-label⟩}{⟨new-options⟩}
```

This removes the original options and replaces them with ⟨*new-options*⟩. If you want to (locally) append to the existing options, use:

```
\mglsAddOptions{⟨multi-label⟩}{⟨new-options⟩}
```

Note that `\multiglossaryentry` doesn't make any adjustments to the component entries. You will need to use the `parent` key when you define the entries if you want a hierarchical structure in your glossary. (See the example in §7.1.1.)

If you don't want the other elements in the glossary, you can suppress the indexing with `indexothers=false` (§7.9.1) or put them in an ignored glossary. For example:

```
\newignoredglossary{common}
\newabbreviation[type={common}]{clostridium}{C.}
{Clostridium}
```

The ⟨*multi-label*⟩ can't be used in commands like `\gls` since the label refers to a set of entry labels not to an individual entry. Similarly, an individual entry label can't be used in commands like `\mgls`. It is possible (although potentially confusing) to use the same label for a multi-entry as for an individual entry (see the example in §7.1.6). Context will determine which is meant, except in the case of the cross-referencing fields (`see`, `seealso` and `alias`) where the cross-referenced label will first be tested if it's a known multi-entry label.

If you don't want to have to keep track of which labels refer to multi-entries and which refer to individual entries you can use:

```
\GlsXtrMglsOrGls{⟨mgls cs⟩}{⟨gls cs⟩}⟨modifier⟩[⟨options⟩]
{⟨label⟩}[⟨insert⟩]
```

where ⟨*mgls cs*⟩ is the `\mgls`-like command to use if ⟨*label*⟩ has been defined as a multi-entry and ⟨*gls cs*⟩ is the `\gls`-like or `\glstext`-like command to use otherwise. The ⟨*modifier*⟩ may be omitted, otherwise it's the modifier that may be used with `\mgls` or `\gls` (asterisk `*`, plus `+` or the token identified with `\GlsXtrSetAltModifier`). The modifier and remaining options are passed to the relevant command (⟨*mgls cs*⟩ or ⟨*gls cs*⟩).

You may prefer to define your own shortcut commands for common combinations. For example, (assuming these commands haven't already been defined by the `shortcuts` option):

```
\newcommand{\ac}{\GlsXtrMglsOrGls{\mgls}{\gls}}
\newcommand{\acp}{\GlsXtrMglsOrGls{\mglsmainpl}
{\glspl}}
\newcommand{\Ac}{\GlsXtrMglsOrGls{\Mgls}{\Gls}}
\newcommand{\Acp}{\GlsXtrMglsOrGls{\Mglsmainpl}
{\Glspl}}
```

7.1. Examples

7.1.1. Example: Hierarchical

Bacteria names are represented by the genus (for example, Clostridium) followed by the species (for example, botulinum). Example 126 has the genus as a parent of the species.

126

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage[stylemods=bookindex,style=bookindex]
{glossaries-extra}

\makeglossaries

\newcommand{\latinname}[1]{\emph{#1}}
\glssetcategoriesattributes
 {genus,species}% categories
 {textformat,glossnamefont}% attributes
 {latinname}

\setabbreviationstyle[genus]
 {long-only-short-only-desc}
\newabbreviation
```

```
 [category={genus},description={}]
 {clostridium}{C.}{Clostridium}

\newglossaryentry{botulinum}{name={botulinum},
  category={species},
  description={},parent={clostridium}}
\newglossaryentry{perfringens}{name={perfringens},
  category={species},description={},
  parent={clostridium}}
\newglossaryentry{tetani}{name={tetani},
  category={species},
  description={},parent={clostridium}}

\multiglossaryentry{cbot}{clostridium,botulinum}
\multiglossaryentry{cperf}{clostridium,perfringens}
\multiglossaryentry{ctet}{clostridium,tetani}

\multiglossaryentrysetup{indexothers=false,hyper=
allmain}

\begin{document}
First use: \mgls{cbot}, \mgls{cperf}, \mgls{ctet}.

Next use: \mgls{cbot}, \mgls{cperf}, \mgls{ctet}.
\printglossaries
\end{document}
```

This suppresses the indexing of the non-main elements (in this case, the genus). However the genus is included in the glossary (without a location list) because it's the parent of the species (which are indexed).

Example 126: Multi-entries: hierarchical

First use: *Clostridium botulinum*, *C. perfringens*, *C. tetani*.
Next use: *C. botulinum*, *C. perfringens*, *C. tetani*.

Glossary

C

Clostridium
botulinum 1
perfringens 1
tetani 1

The hyper=allmain option makes the entire content of each \mgls a hyperlink to the main entry in the glossary.

7.1.2. Example: Suffix

Example 127 is a minor modification of Example 126. In this case the multi-entries are defined with a suffix: 127

```
\multiglossaryentry[firstsuffix=botulism]{cbot}
{clostridium,botulinum}
\multiglossaryentry[firstsuffix=gas gangrene]{cperf}
{clostridium,perfringens}
\multiglossaryentry[firstsuffix=tetanus]{ctet}
{clostridium,tetani}
```

The rest of the document is as for Example 126 in §7.1.1.

Example 127: Multi-entries: hierarchical with first-use suffix

First use: *Clostridium botulinum* (botulism), *C. perfringens* (gas gangrene), *C. tetani* (tetanus).
Next use: *C. botulinum*, *C. perfringens*, *C. tetani*.

Glossary

C

Clostridium
botulinum 1
perfringens 1
tetani 1

7.1.3. Example: Category Suffix

Example 128 is an alternative to Example 127. Instead of storing the extra information in the firstsuffix key, the information is stored in the user1 key of the last element (the species). A category suffix is used to look up the field and append it. 128

```
\newglossaryentry{botulinum}{name={botulinum},
  category={species},
  user1={botulism},
```

```
  description={},parent={clostridium}}
\newglossaryentry{perfringens}{name={perfringens},
  category={species},
  user1={gas gangrene},
  description={},parent={clostridium}}
\newglossaryentry{tetani}{name={tetani},
  category={species},
  user1={tetanus},
  description={},parent={clostridium}}

\mglsdefcategorysuffix{bacteria}{%
 \mglsisfirstuse
 {% only on first use:
  \glsxtrifhasfield{useri}{\mglslastelementlabel}%
   { (\glscurrentfieldvalue)}
   {}%
 }%
 {}%
}

\multiglossaryentry[category=bacteria]{cbot}
{clostridium,botulinum}
\multiglossaryentry[category=bacteria]{cperf}
{clostridium,perfringens}
\multiglossaryentry[category=bacteria]{ctet}
{clostridium,tetani}
```

The result is the same as for Example 127.

Example 128: Multi-entries: hierarchical with category suffix

First use: *Clostridium botulinum* (botulism), *C. perfringens* (gas gangrene), *C. tetani* (tetanus).
Next use: *C. botulinum*, *C. perfringens*, *C. tetani*.

Glossary

C

Clostridium
botulinum 1
perfringens 1
tetani 1

7.1.4. Example: Separators

Example 129 modifies Example 126 (from §7.1.1) so that the species are also abbreviations. In this case, the separators are modified to suppress the space (`\relax`) if both the genus and species are abbreviated, or to use a non-breaking space (~) between the genus short form (shown on subsequent use) and the species long form (shown on first use). If the genus is showing the long form (first use) then a normal space is used. 129

Note that the separator attributes apply to the category of the element before the separator (not to the multi-entry category).

```
\glssetcategoryattribute{genus}
 {combinedfirstsepfirst}{\space}
\glssetcategoryattribute{genus}
 {combinedfirstsep}{\space}
\glssetcategoryattribute{genus}
 {combinedsepfirst}{~}
\glssetcategoryattribute{genus}{combinedsep}{\relax}

\setabbreviationstyle[species]
 {long-only-short-only-desc}

\newabbreviation[category={species},
  description={},parent={clostridium}]{botulinum}
{bot.}{botulinum}
\newabbreviation[category={species},
  description={},parent={clostridium}]{perfringens}
{per.}{perfringens}
\newabbreviation[category={species},
  description={},parent={clostridium}]{tetani}{tet.}
{tetani}
```

This will cause a double dot at the end of the second sentence, which can be suppressed using the `discardperiod` and `retainfirstuseperiod` attributes.

```
\glssetcategoriesattributes{species}
 {discardperiod,retainfirstuseperiod}{true}
```

This works because the final element's post-link hook is transferred to the multi-entry post-link hook, which can detect the sentence terminating period. If the post-link hook settings are changed, for example, to `postlinks=all,mpostlink=false` then the feature won't work as the final element's post-link hook can't detect the period (because `\gls` is embedded too deeply inside the internal workings of `\mgls`).

Example 129: Multi-entries: separators

First use: *Clostridium botulinum*, *C. perfringens*, *C. tetani*.
Next use: *C.bot.*, *C.per.*, *C.tet.*

Glossary

C

Clostridium
botulinum 1

perfringens 1
tetani 1

7.1.5. Example: Skipping Elements (Fragment Element)

Example 130 is an alternative way of dealing with nested links (see §5.4).

130

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage[stylemods,style=long]{glossaries-extra}

\makeglossaries

\setabbreviationstyle{long-short-sc}
\newabbreviation{ssi}{ssi}{server-side includes}
\newabbreviation{html}{html}
{hypertext markup language}

\setabbreviationstyle[combinedabbrv]
 {long-only-short-sc-only}
\newabbreviation
 [category={combinedabbrv},
  description={\glsxtrshort{ssi} enabled
   \glsxtrshort{html}}]
 {shtml-frag}{shtml}{enabled}

\glssetcategoryattribute
 {combinedabbrv}{multioptions}
 {usedskipothers,
  firstsuffix={\glsxtrshort{\mglslastmainlabel}}
```

```
 }

\multiglossaryentry
 [category=combinedabbrv]
 {shtml}[shtml-frag]{ssi,shtml-frag,html}

\begin{document}
Individual elements first use: \gls{ssi} and
\gls{html}.

Individual elements next use: \gls{ssi} and
\gls{html}.

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

Resetting all\glsresetall\mglsreset{shtml}:

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.
\printglossaries
\end{document}
```

This uses the `multioptions` attribute to skip "other" elements on subsequent use. The problematic abbreviation (SHTML) is defined as a fragment that simply expands to "enabled" on first use. Note that the description has to be supplied for the glossary.

Example 130: Multi-entries: skipping elements

Individual elements first use: server-side includes (SSI) and hypertext markup language (HTML).
Individual elements next use: SSI and HTML.
Multi-entry first use: SSI enabled HTML (SHTML).
Multi-entry next use: SHTML.
Resetting all:
Multi-entry first use: server-side includes (SSI) enabled hypertext markup language (HTML) (SHTML).
Multi-entry next use: SHTML.
Individual elements: SSI and HTML.

Glossary

HTML	hypertext markup language 1
SHTML	SSI enabled HTML 1
SSI	server-side includes 1

The key difference here from Example 96, which uses `\glsps`, is that the individual elements hyperlink to their respective entries in the glossary on first use of `\mgls`.

The problem is that with the `colorlinks` package option, it's not obvious where the hyperlinks start and end. The suffix (SHTML) for the multi-entry first use will hyperlink to the "shtml" entry in the glossary, so the "enabled" hyperlink is redundant. The simplest fix for this is to add `hyper=notmainfirst` to the option list, which will prevent "enabled" from being a hyperlink.

Another problem occurs where `\mgls` is used before the individual elements are used, which leads to their full expansion with a confusing amount of parentheses. A simple solution is to use the option `mglsopts=unsetothers`, which will unset the other (not-main) elements first. This can be localised with `presetlocal` but `\gls` will then unset the first use flag globally, which means that the other elements won't show the full form when they are first used on their own after `\mgls`. This can be switched to a local unset with `others=local`.

Example 131 adapts the previous example to incorporate these options:

131

```
\glssetcategoryattribute
 {combinedabbrv}{multioptions}
 {hyper=notmainfirst,
  mglsopts={presetlocal,unsetothers,others=local},
```

```
  usedskipothers,
  firstsuffix=\glsxtrshort{\mglslastmainlabel}
 }
```

Example 131: Multi-entries: skipping elements (unsetting others)

Individual elements first use: server-side includes (SSI) and hypertext markup language (HTML).
Individual elements next use: SSI and HTML.
Multi-entry first use: SSI enabled HTML (SHTML).
Multi-entry next use: SHTML.
Resetting all:
Multi-entry first use: SSI enabled HTML (SHTML).
Multi-entry next use: SHTML.
Individual elements: server-side includes (SSI) and hypertext markup language (HTML).

Glossary

HTML	hypertext markup language 1
SHTML	SSI enabled HTML 1
SSI	server-side includes 1

This method still has two main drawbacks: the description must be added manually and the long form can't be accessed with `\glsxtrlong`. The next example provides an alternative approach.

7.1.6. Example: Skipping Elements (Prefix and Post-Link Hooks)

Example 132 is a modified version of the previous example. In this case, the main element isn't a fragment and also happens to have the same label as the multi-entry set. (`\mgls{shtml}` references the multi-entry label and `\gls{shtml}` references the individual entry.) 132

In this case, the nested parts are marked up with custom commands:

```
\newrobustcmd{\combinedpre}[1]{\glsps{#1}}
\newrobustcmd{\combinedpost}[1]{\glsps{#1}}

\newabbreviation{shtml}{shtml}
```

```
{{}\combinedpre{ssi} enabled \combinedpost{html}}
```

This means that it's no longer necessary to manually insert the description and the long form can be accessed as usual with `\glsxtrshort{shtml}`. Note that it is necessary to define the custom commands robustly otherwise they will need to be protected against premature expansion:

```
\newcommand{\combinedpre}[1]{\glsps{#1}}
\newcommand{\combinedpost}[1]{\glsps{#1}}

\newabbreviation{shtml}{shtml}
{{}\protect\combinedpre{ssi}
enabled \protect\combinedpost{html}}
```

In both cases, an initial empty group is added to guard against any sentence case commands, such as `\Glsxtrlong`.

The abbreviations all use the long-postshort-sc-user style, which places the short form in the post-link hook on first use. The `\gls` post-link hook for the main element can be transferred to the `\mgls` post-link using:

```
mpostlinkelement=main
```

All elements have their individual post-link hooks suppressed by default. As in the previous example, the other elements can be skipped on subsequent use:

```
usedskipothers
```

Within `\mgls`, the nested content needs to be suppressed, which can be done by redefining the custom commands. This can be done in the multi-entry prefix. Since the entire content of `\mgls` (except for the final multi-entry post-link hook) occurs inside a group, this redefinition will be localised.

The complete document is as follows:

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage[stylemods,style=long]{glossaries-extra}

\makeglossaries
```

```
\setabbreviationstyle{long-postshort-sc-user}

\newabbreviation{ssi}{ssi}{server-side includes}
\newabbreviation{html}{html}
{hypertext markup language}

\newrobustcmd{\combinedpre}[1]{\glsps{#1}}
\newrobustcmd{\combinedpost}[1]{\glsps{#1}}

\newabbreviation{shtml}{shtml}
{{}\combinedpre{ssi} enabled \combinedpost{html}}

\glssetcategoryattribute
 {combinedabbrv}{multioptions}
 {mpostlinkelement=main,
  usedskipothers}

\multiglossaryentry
 [category=combinedabbrv]
 {shtml}[shtml]{ssi,shtml,html}

\mglsdefcategoryprefix{combinedabbrv}{%
 \renewcommand{\combinedpre}[1]{\ignorespaces}%
 \renewcommand{\combinedpost}[1]{\unskip}%
}

\begin{document}
Individual elements first use: \gls{ssi} and
\gls{html}.

Individual elements next use: \gls{ssi} and
\gls{html}.

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

Individual entry first use: \gls{shtml}.

Resetting all\glsresetall\mglsresetall:

Multi-entry first use: \mgls{shtml}.
```

```
Multi-entry next use: \mgls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.

Resetting all\glsresetall\mglsresetall:

Individual entry first use: \gls{shtml}.

Multi-entry first use: \mgls{shtml}. (Wrong!)

\printglossaries
\end{document}
```

Example 132: Multi-entries: skipping elements (prefix and post-link hooks)

Individual elements first use: server-side includes (SSI) and hypertext markup language (HTML).
Individual elements next use: SSI and HTML.
Multi-entry first use: SSI enabled HTML (SHTML).
Multi-entry next use: SHTML.
Individual entry first use: SHTML.
Resetting all:
Multi-entry first use: server-side includes enabled hypertext markup language (SHTML).
Multi-entry next use: SHTML.
Individual elements: SSI and HTML.
Resetting all:
Individual entry first use: SSI enabled HTML (SHTML).
Multi-entry first use: server-side includes SHTML hypertext markup language. (Wrong!)

Glossary

HTML	hypertext markup language 1
SHTML	SSI enabled HTML 1
SSI	server-side includes 1

Note the last two paragraphs, which highlights what happens if `\gls{shtml}` is used before `\mgls{shtml}` when neither of the other elements (`ssi` and `html`) have been used. The

final instance of \mgls has produced the wrong result. This is because it's the first use of the multi-entry shtml but not the first use of the individual entry shtml.

One way around this is to modify the prefix to ensure that the main element's first use flag matches the multi-entry's first use flag:

```
\mglsdefcategoryprefix{combinedabbrv}{%
 \renewcommand{\combinedpre}[1]{\ignorespaces}%
 \renewcommand{\combinedpost}[1]{\unskip}%
 \mglsisfirstuse
 {\glslocalreset{\mglscurrentmainlabel}}%
 {\glslocalunset{\mglscurrentmainlabel}}%
}
```

The showtargets=annoteleft option can be used to mark up the links with the targets. For example, the first instance of \mgls{shtml} will show as:

Multi-entry first use: [glo:ssi]▹SSI◃ [glo:shtml]▹enabled◃ [glo:html]▹HTML◃ (SHTML).

Each entry has an individual hyperlink to its own glossary item, which may be confusing. This can be made clearer by suppressing the main element link on first use with:

```
hyper=notmainfirst
```

(as in the previous example), and adjusting the abbreviation style so that the parenthetical content in the post-link hook has a hyperlink:

```
\renewcommand*{\glsxtruserparen}[2]{%
  \glsxtrfullsep{#2}%
  \glsxtrparen
   {\glshyperlink[#1]{#2}%
    \ifglshasfield{\glsxtruserfield}{#2}{,
      \glscurrentfieldvalue}{}%
   }%
}
```

The remaining problem is how to deal with the possibility that \mgls{shtml} may come before the first use of the other elements. For example:

```
Multi-entry first use: \mgls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.
```

This leads to:

Multi-entry first use: server-side includes enabled hypertext markup language (SHTML).
Individual elements: SSI and HTML.

This means that the abbreviations SSI and HTML aren't explained in the document text. One way around this is to only locally unset the other element first use flags:

```
\glssetcategoryattribute
 {combinedabbrv}{multioptions}
 {hyper=notmainfirst,
  mpostlinkelement=main,
  usedskipothers,
  mglsopts={others=local}
 }
```

With the above setting, the following:

```
\glsresetall\mglsresetall

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.
```

will now produce:

Multi-entry first use: server-side includes enabled hypertext markup language (SHTML).
Multi-entry next use: SHTML.
Individual elements: server-side includes (SSI) and hypertext markup language (HTML).

7.2. Main and Other Elements

The list of labels provided in the final argument of `\multiglossaryentry` consists of a main element and all the other elements. If the main element isn't identified in the optional argument, it's assumed to be the final element in the list.

The main element allows you to determine which target to use if you want the entire content of `\mgls` to be a single hyperlink. You can also use the settings described in §7.9 to only index the main element.

You can change the main element using: `\mglsSetMain` The new main label provided in the second argument must be in the list corresponding to ⟨*multi-label*⟩. This change is locally applied to the current scope. Note that if you are using `bib2gls`, this change in the document can't be detected.

The main element can also be used to identify which element should be displayed in the plural with `\mglsmainpl`. For example:

```
\newglossaryentry{great}{name={great},
 description={}}
\newglossaryentry{little}{name={little},
 description={}}
\newglossaryentry{grebe}{name={grebe},
 description={}}

\multiglossaryentry{greatgrebe}{great,grebe}
\multiglossaryentry{littlegrebe}{little,grebe}
```

In the above, two multi-entries are defined: `greatgrebe` and `littlegrebe`. In both cases the main element is `grebe` (the last element). Using `\mglspl` will show the plural for all elements, but using `\mglsmainpl` will only use the plural for the main element (grebe). For example:

```
Plural all: \mglspl{greatgrebe},
\mglspl{littlegrebe}.
Plural main: \mglsmainpl{greatgrebe},
\mglsmainpl{littlegrebe}.
```

produces:

Plural all: greats grebes, littles grebes. Plural main: great grebes, little grebes.

7.3. Prefixes and Suffixes

A multi-entry may have associated prefixes and suffixes. These are scoped and are placed outside of the hyperlinks and encapsulating commands. They are not affected by case-changing commands, such as `\Mgls`. If you want a prefix to obey case-changing, use the `\mpgls`-like commands instead (§7.11.4).

The prefix is inserted with:

```
\mglsprefix
```

The default definition is:

```
\newcommand*{\mglsprefix}{%
 \ifdefempty\mglscurrentcategory
 {\mglscurrentprefix}%
 {%
   \mglshascategoryprefix{\mglscurrentcategory}%
   {\mglsusecategoryprefix{\mglscurrentcategory}}%
   {\mglscurrentprefix}%
 }%
}
```

This will insert the current prefix unless there is prefix command associated with the current category.

The suffix is inserted with:

```
\mglssuffix
```

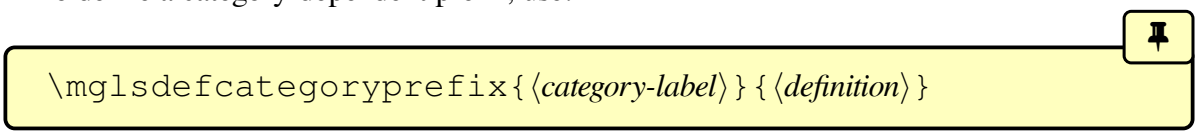
This command is defined as follows:

```
\newcommand*{\mglssuffix}{%
 \ifdefempty\mglscurrentcategory
 {\ifdefempty\mglscurrentsuffix{}{\space
   (\mglscurrentsuffix)}}%
 {%
   \mglshascategorysuffix{\mglscurrentcategory}%
   {\mglsusecategorysuffix{\mglscurrentcategory}}%
   {\ifdefempty{\mglscurrentsuffix}{}{\space
     (\mglscurrentsuffix)}}%
 }%
}
```

If there is a suffix associated with the current category, that will be used, otherwise if the current suffix isn't empty this inserts a space followed by the current suffix in parentheses. You can access the label of the last (non-skipped) element with `\mglslastelementlabel`.

Note that in both cases the category corresponds to the multi-entry category (see §7.8).

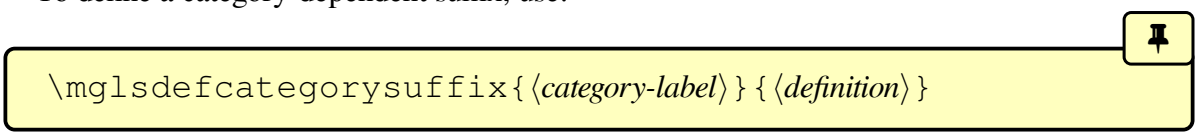
To define a category-dependent prefix, use:



`\mglsdefcategoryprefix{`⟨*category-label*⟩`}{`⟨*definition*⟩`}`

You can reference the current prefix with `\mglscurrentprefix` within ⟨*definition*⟩.

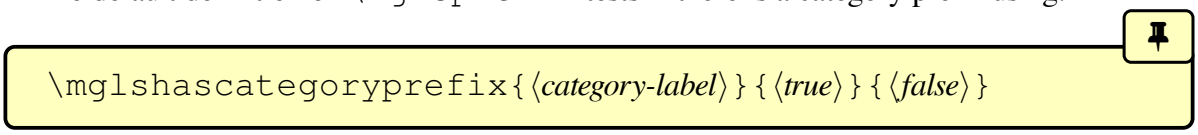
To define a category-dependent suffix, use:



`\mglsdefcategorysuffix{`⟨*category-label*⟩`}{`⟨*definition*⟩`}`

You can reference the current suffix with `\mglscurrentsuffix` within ⟨*definition*⟩.

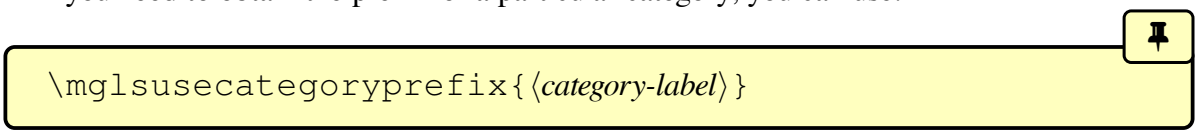
The default definition of `\mglsprefix` tests if there is a category prefix using:



`\mglshascategoryprefix{`⟨*category-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This does ⟨*true*⟩ if a prefix has been assigned to the given category, otherwise it does ⟨*false*⟩.

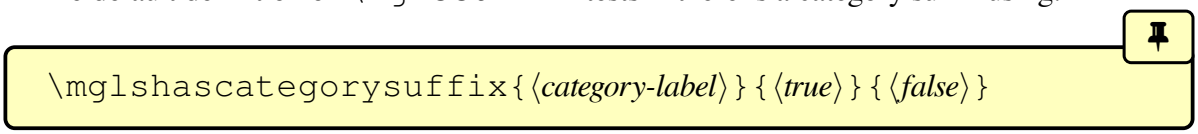
If you need to obtain the prefix for a particular category, you can use:



`\mglsusecategoryprefix{`⟨*category-label*⟩`}`

This expands to the prefix, if set, for the given category or to nothing otherwise.

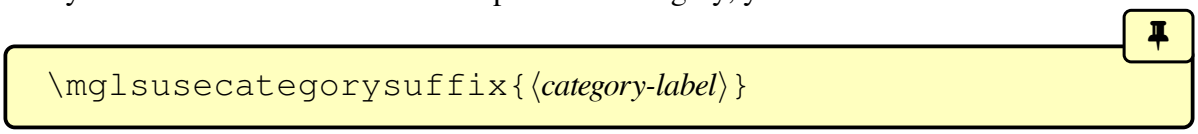
The default definition of `\mglssuffix` tests if there is a category suffix using:



`\mglshascategorysuffix{`⟨*category-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This does ⟨*true*⟩ if a suffix has been assigned to the given category, otherwise it does ⟨*false*⟩.

If you need to obtain the suffix for a particular category, you can use:



`\mglsusecategorysuffix{`⟨*category-label*⟩`}`

This expands to the suffix, if set, for the given category or to nothing otherwise.

The current prefix `\mglscurrentprefix` and `\mglscurrentsuffix` are obtained as follows:

- if this is the first use of the multi-entry (§7.7) then the ⟨*prefix*⟩ is set to the value of the `firstprefix` option and the ⟨*suffix*⟩ is set to the value of the `firstsuffix` option;

- otherwise the ⟨*prefix*⟩ is set to the value of the `usedprefix` option and the ⟨*suffix*⟩ is set to the value of the `usedsuffix` option.

The prefix and suffix (if set) are placed outside of the hyperlink and text formatting encapsulator. They are not affected by case-changing commands such as `\Mgls` or `\MGLS`.

For example:

```
\setabbreviationstyle{long-only-short-only}
\newabbreviation{clostridium}{C.}{Clostridium}
\newglossaryentry{botulinum}{name={botulinum},
 description={}}

\multiglossaryentry[firstsuffix=botulism]
 {cbot}{clostridium,botulinum}
```

On first use, this produces (assuming the "clostridium" element hasn't been used previously):

Clostridium botulinum (botulism).

On subsequent use, this produces:

C. botulinum.

7.4. Separators

The separators between each instance of `\gls` are given by the following commands, which all take two arguments. The first argument is the label of the previous element. The second argument is the label of the following element.

`\glscombinedsep{`⟨*prev label*⟩`}{`⟨*next label*⟩`}`

This is inserted between two entries that have both been marked as used. The default definition is:

```
\newcommand*{\glscombinedsep}[2]{%
  \glshasattribute{#1}{combinedsep}%
  {\glsgetattribute{#1}{combinedsep}}%
  { }%
}
```

This will use the `combinedsep` attribute for the ⟨*prev label*⟩'s category, if set. Otherwise this just does a space. Note that this ignores the second argument.

`\glscombinedfirstsep{`⟨*prev label*⟩`}{`⟨*next label*⟩`}`

This is inserted between two entries where only the next entry has been marked as used. The default definition is:

```
\newcommand*{\glscombinedfirstsep}[2]{%
 \glshasattribute{#1}{combinedfirstsep}%
 {\glsgetattribute{#1}{combinedfirstsep}}%
 {\glscombinedsep{#1}{#2}}%
}
```

This will use the `combinedfirstsep` attribute for ⟨*prev label*⟩'s category, if set. If that attribute isn't set, `\glscombinedsep` is used.

`\glscombinedsepfirst{`⟨*prev label*⟩`}{`⟨*next label*⟩`}`

This is inserted between two entries where only the previous entry has been marked as used. The default definition is:

```
\newcommand*{\glscombinedsepfirst}[2]{%
 \glshasattribute{#1}{combinedsepfirst}%
 {\glsgetattribute{#1}{combinedsepfirst}}%
 {\glscombinedsep{#1}{#2}}%
}
```

This will use the `combinedsepfirst` attribute for ⟨*prev label*⟩'s category, if set. If that attribute isn't set, `\glscombinedsep` is used.

`\glscombinedfirstsepfirst{`⟨*prev label*⟩`}{`⟨*next label*⟩`}`

This is inserted between two entries where both have been marked as used. The default definition is:

```
\newcommand*{\glscombinedfirstsepfirst}[2]{%
 \glshasattribute{#1}{combinedfirstsepfirst}%
 {\glsgetattribute{#1}{combinedfirstsepfirst}}%
 {\glscombinedsep{#1}{#2}}%
}
```

This will use the `combinedfirstsepfirst` attribute for ⟨*prev label*⟩'s category, if set. If that attribute isn't set, `\glscombinedsep` is used.

These commands may be redefined as required. For example, to have no space between two elements that have both been marked as used and are both abbreviations (disregarding category attributes):

```
\renewcommand*{\glscombinedfirstsepfirst}[2]{%
 \ifglshasshort{#1}%
  {\ifglshasshort{#2}{}{\space}}%
  {\space}%
}
```

There are some commands for redefining the above separators to common combinations.

```
\glssetcombinedsepabbrvnbsp
```

This does the following:

```
\renewcommand*{\glscombinedsep}[2]{%
 \glshasattribute{#1}{combinedsep}%
 {\glsgetattribute{#1}{combinedsep}}%
 {\ifglshasshort{#1}{~}{ }}%
}%
\renewcommand*{\glscombinedsepfirst}[2]{%
 \glshasattribute{#1}{combinedsepfirst}%
 {\glsgetattribute{#1}{combinedsepfirst}}%
 {\ifglshasshort{#1}{~}{ }}%
}%
\renewcommand*{\glscombinedfirstsep}[2]{%
 \glshasattribute{#1}{combinedfirstsep}%
```

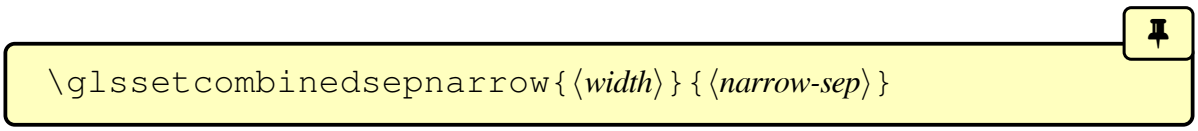
```
 {\glsgetattribute{#1}{combinedfirstsep}}%
 { }%
}%
\renewcommand*{\glscombinedfirstsepfirst}[2]{%
  \glshasattribute{#1}{combinedfirstsepfirst}%
  {\glsgetattribute{#1}{combinedfirstsepfirst}}%
  { }%
}
```

This uses a non-breaking space (~) following an abbreviation (that has already been marked as used). Note that if the associated attributes are set the commands will behave according to the attribute.

```
\glssetcombinedsepabbrvnone
```

```
\renewcommand*{\glscombinedsep}[2]{%
 \glshasattribute{#1}{combinedsep}%
 {\glsgetattribute{#1}{combinedsep}}%
 {\ifglshasshort{#1}{}{\ifglshasshort{#2}{}{ }}}%
}%
\renewcommand*{\glscombinedsepfirst}[2]{%
 \glshasattribute{#1}{combinedsepfirst}%
 {\glsgetattribute{#1}{combinedsepfirst}}%
 {\ifglshasshort{#1}{}{ }}%
}%
\renewcommand*{\glscombinedfirstsep}[2]{%
 \glshasattribute{#1}{combinedfirstsep}%
 {\glsgetattribute{#1}{combinedfirstsep}}%
 {\ifglshasshort{#2}{}{ }}%
}%
\renewcommand*{\glscombinedfirstsepfirst}[2]{%
 \glshasattribute{#1}{combinedfirstsepfirst}%
 {\glsgetattribute{#1}{combinedfirstsepfirst}}%
 { }%
}
```

This does nothing if either element are abbreviations that have already been used. Note that if the associated attributes are set the commands will behave according to the attribute.



\glssetcombinedsepnarrow{⟨width⟩}{⟨narrow-sep⟩}

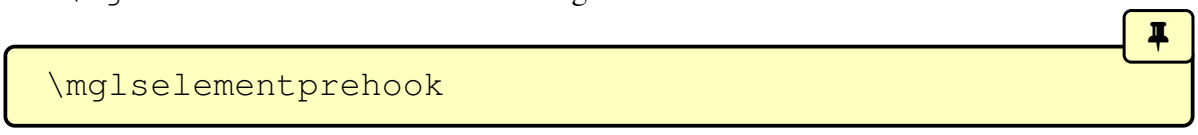
This is rather more complicated as it measures a field value and uses ⟨*narrow-sep*⟩ if the width is less than ⟨*width*⟩. The field value is determined as follows:

- on first use the `long` field is used if it is set otherwise the `first` field is used;
- otherwise the `short` field is used if it is set otherwise the `text` field is used;

Note that this doesn't take into account fonts, hooks, abbreviation styles or plural forms (e.g. `\mglspl`) or other field references (e.g. `\mglsname`). If the associated attributes are set the commands will behave according to the attribute.

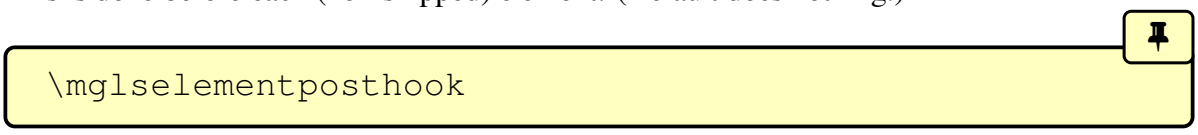
7.5. `\mgls` Element Hooks

The `\mgls`-like commands use the following hooks:



\mglselementprehook

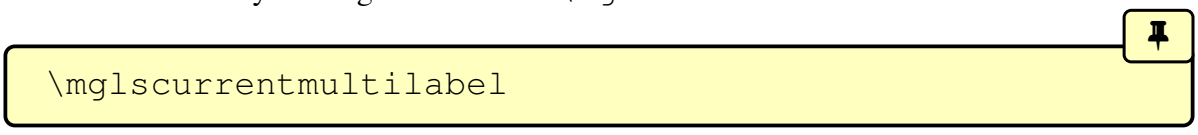
This is done before each (non-skipped) element. (Default does nothing.)



\mglselementposthook

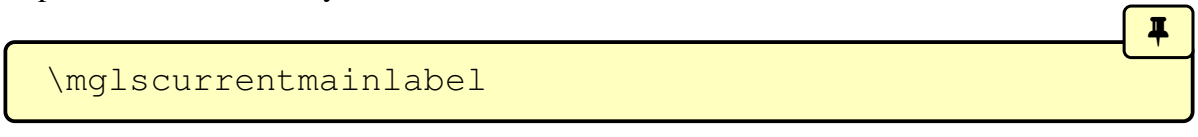
This is done after each (non-skipped) element. (Default does nothing.) Note that this is different from the normal entry post-link hook `\glspostlinkhook`. If the individual entry post-link hook is enabled (see the `postlinks` key in §7.9), this will go before `\mglselement-posthook`.

The definitions of the following commands are scoped within the `\mgls`-like commands so they can't be accessed elsewhere (including in the post-link hook, see §7.6). They may be used in the above hooks or in the separator commands (described in §7.4) or in the command used to encapsulate the entire content. They can also be used in the post-link hook (see §5.5) to determine if an entry is being used within a `\mgls`-like command.



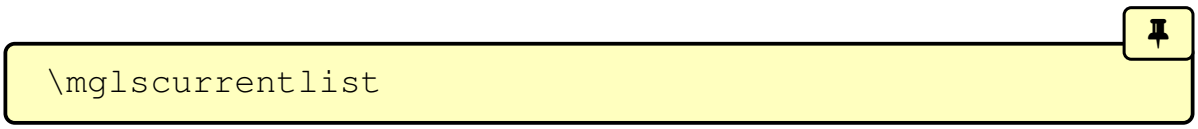
\mglscurrentmultilabel

Expands to the multi-entry label.



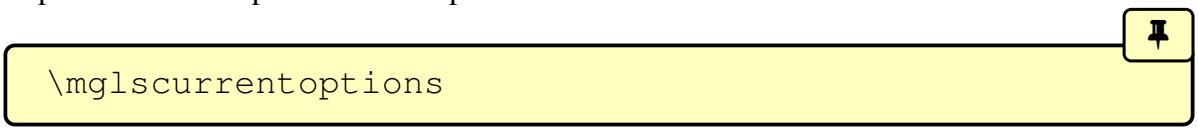
\mglscurrentmainlabel

Expands to the label of the main element.



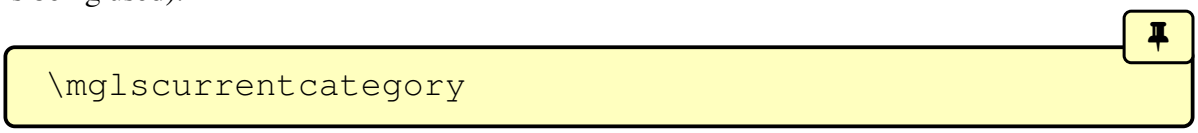
```
\mglscurrentlist
```

Expands to the complete comma-separated list of elements.



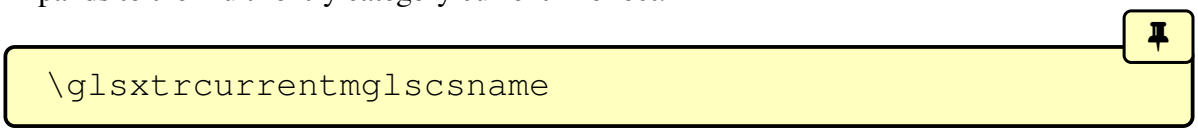
```
\mglscurrentoptions
```

Expands to the options used when the multi-entry was defined. This doesn't include options set with `\multiglossaryentrysetup` or those passed to `\mgls` (or whichever variant is being used).



```
\mglscurrentcategory
```

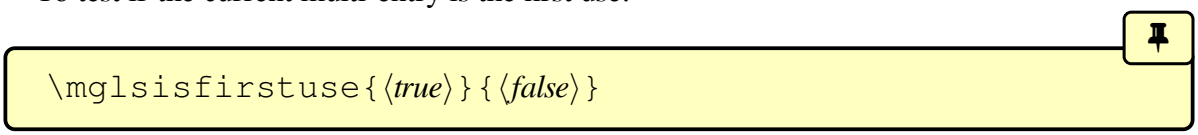
Expands to the multi-entry category current in effect.



```
\glsxtrcurrentmglscsname
```

Expands to the control sequence name of the calling command (for example, `mgls` or `mglspl`).

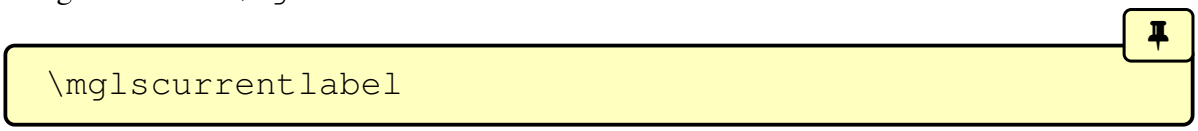
To test if the current multi-entry is the first use:



```
\mglsisfirstuse{⟨true⟩}{⟨false⟩}
```

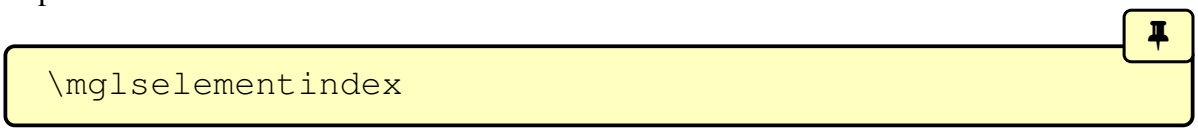
This does ⟨*true*⟩ if this is the first use otherwise it does ⟨*false*⟩. Note that this applies to the multi-entry first use flag not the first use flags of the individual elements.

At each iteration of the loop over the element list, the following commands are set, which can be accessed in hooks such as `\mglselementprehook` or in hooks used by the underlying `\gls` etc commands. For example, if `\mglscurrentlabel` is defined then `\gls` is being used inside `\mgls`.



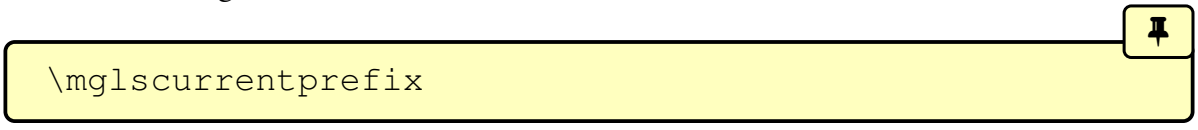
```
\mglscurrentlabel
```

Expands to the current element label.



```
\mglselementindex
```

This is a count register that is set to the element index.



```
\mglscurrentprefix
```

Expands to the current multi-entry prefix.

```
\mglscurrentsuffix
```

Expands to the current multi-entry suffix.

```
\mglsiflast{⟨true⟩}{⟨false⟩}
```

If this is the last iteration, does ⟨*true*⟩ otherwise does ⟨*false*⟩. This takes the skip options into account, so the last iteration may not necessarily be when the element index is equal to the total number of elements.

7.6. Post-Link Hook

There is a hook that occurs at the end the `\mgls`-like commands according to the `mpostlink` setting (see §7.9). The hook used depends on the `mpostlinkelement` option. These hooks can't access the commands described in §7.5 as the hook occurs outside of the scope in which they are defined.

The `mpostlinkelement=custom` option uses:

```
\mglscustompostlinkhook
```

This does nothing by default.

The `mpostlinkelement=last` option uses:

```
\mglslastelementpostlinkhook
```

which emulates the post-link hook of the last element.

The `mpostlinkelement=main` option uses:

```
\mglslastmainpostlinkhook
```

which emulates the post-link hook of the main element.

The default settings `postlinks=none, mpostlink=true, and mpostlinkelement=last` will suppress the individual element post-link hooks (`\glspostlinkhook`) and do the multi-entry post-link hook for the last element (`\mglslastelementpostlinkhook`).

If you have the final element's post-link hook enabled and the multi-entry post-link hook enabled (for example, `postlinks=all, mpostlink=true, mpostlinkelement=last`), the final element's post-link hook will be done twice. Similarly for the main element with `postlinks=all, mpostlink=true, mpostlinkelement=main`.

The following commands are available for use in these hooks and may also be used in the definition of `\mglssuffix`.

```
\mglslastmultilabel
```

Expands to the multi-entry label.

```
\mglslastcategory
```

Expands to the multi-entry category (see §7.8). This will be empty if no category was assigned.

```
\mglswasfirstuse{⟨true⟩}{⟨false⟩}
```

If that was the first use of the multi-entry (see §7.7) this does ⟨*true*⟩ otherwise it does ⟨*false*⟩.

7.6.1. Last Element

The following commands relate to the last element.

```
\mglslastelementlabel
```

Expands to the label of the last non-skipped element. If all elements were skipped or if the multi-entry wasn't defined, this will be empty.

Test if the last element was skipped:

```
\mglsiflastelementskipped{⟨true⟩}{⟨false⟩}
```

If the last element was skipped this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. If all elements were skipped or if the multi-entry wasn't defined, this will do ⟨*true*⟩.

Test if the last element was its first use:

```
\mglsiflastelementwasfirstuse{⟨true⟩}{⟨false⟩}
```

If the last non-skipped element was used for the first time this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. (Corresponds to `\glsxtrifwasfirstuse`.) If all elements were skipped or if the multi-entry wasn't defined, this will do ⟨*true*⟩.

Test if the last element was plural:

```
\mglsiflastelementwasplural{⟨true⟩}{⟨false⟩}
```

If the last non-skipped element had the plural form displayed, this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. (Corresponds to `\glsifplural`.) If all elements were skipped or if the multi-entry wasn't defined, this will do ⟨*false*⟩.

Test if the last element was had any case-changing applied:

```
\mglsiflastelementcapscase{⟨no-change⟩}{⟨firstuc⟩}{⟨all caps⟩}
```

Corresponds to `\glscapscase` of the last non-skipped element. If all elements were skipped or if the multi-entry wasn't defined, this will do ⟨*no-change*⟩.

7.6.2. Main Element

The following commands relate to the main element.

```
\mglslastmainlabel
```

Expands to the label of the main element from the multi-entry that was just referenced. If the main element was skipped or if the multi-entry wasn't defined, this will be empty. If this is the same as `\mglslastelementlabel` then the main element was the last element.

Test if the main element was skipped:

```
\mglsiflastmainskipped{⟨true⟩}{⟨false⟩}
```

If the main element from the multi-entry that was just referenced was skipped this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. If the multi-entry wasn't defined, this will do ⟨*true*⟩.

Test if the main element was its first use:

```
\mglsiflastmainwasfirstuse{⟨true⟩}{⟨false⟩}
```

If the main element was used for the first time this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. (Corresponds to `\glsxtrifwasfirstuse`.) If the main element was skipped or if the multi-entry wasn't defined, this will do ⟨*true*⟩.

Test if the main element was plural:

```
\mglsiflastmainwasplural{⟨true⟩}{⟨false⟩}
```

If the main element from the multi-entry that was just referenced had its plural form displayed this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. (Corresponds to `\glsifplural`.) If the main element was skipped or if the multi-entry wasn't defined, this will do ⟨*false*⟩.

Test if the main element was had any case-changing applied:

```
\mglsiflastmaincapscase{⟨no-change⟩}{⟨firstuc⟩}{⟨all caps⟩}
```

Corresponds to `\glscapscase` of the main element from the multi-entry that was just referenced. If the main element was skipped or if the multi-entry wasn't defined, this will do ⟨*no-change*⟩.

7.7. Multi-Entry First Use

Each multi-entry set has an associated first use flag. This is independent of the first use flag associated with the individual entries that make up the set. As with the `\gls`-like commands, `\mgls` unsets this flag. Unlike the `\glstext`-like commands, all the commands described in §7.11 (including commands like `\mglsname`) unset this flag, even if the elements use commands like `\glsname` that don't unset the entry's first use flag.

You can determine whether or not a multi-entry set has been marked as used with:

```
\ifmglsused{⟨multi-label⟩}{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if the given multi-entry has been marked as used, otherwise it does ⟨*false*⟩.

You can (globally) unset the flag (mark the set as having been referenced) with:

```
\mglsunset{⟨multi-label⟩}
```

or reset it with:

```
\mglsreset{⟨multi-label⟩}
```

There are also local versions of these commands:

```
\mglslocalunset{⟨multi-label⟩}
```

which locally unsets the flag and

```
\mglslocalreset{⟨multi-label⟩}
```

which locally resets the flag.

It's also possible to unset or reset all multi-entries.

```
\mglsunsetall
```

Unsets all multi-entries.

\mglsresetall

Resets all multi-entries.

Note that unsetting or resetting any of the individual element first use flags doesn't affect the multi-entry flag. Similarly, unsetting or resetting the multi-entry flag doesn't affect the first use flags of the individual elements.

7.8. Multi-Entry Category

A multi-entry set may have an associated category set using the category key described in §7.9. This isn't set by default, but if it is set the category may have attributes set in the usual way. The multi-entry category is independent of the individual entry categories. The following attribute is recognised by commands like \mgls:

multioptions=⟨*options*⟩

The value are the default options to apply to any multi-entry set with the given category. These can be overridden by the first optional argument of \multiglossaryentry or by the setup key in the first optional argument of commands like \mgls.

combinedfirstsep=⟨*separator*⟩

The separator to use for \glscombinedfirstsep.

combinedfirstsepfirst=⟨*separator*⟩

The separator to use for \glscombinedfirstsepfirst.

combinedsepfirst=⟨*separator*⟩

The separator to use for \glscombinedsepfirst.

combinedsep=⟨*separator*⟩

The separator to use for \glscombinedsep.

Note that you can't access the category or its attributes via the multi-entry label (for example, with \glshasattribute). If you need to access the current multi-entry's category within

any of the \mgls-like hooks (§7.5), you can obtain the category with \mglscurrentcategory and use commands like \glshascategoryattribute.

7.9. Multi-Entry Settings

The settings that govern all multi-entries can be set using:

\multiglossaryentrysetup{⟨*options*⟩}

The ⟨*options*⟩ are the same as for \multiglossaryentry.

Whenever the \mgls-like commands are used, options are applied in the following order:

1. general options identified by \multiglossaryentrysetup;
2. the category key is assigned if it's in the general options, \multiglossaryentry or setup key;
3. multioptions category attribute (if set);
4. any options provided in the first optional argument of \multiglossaryentry;
5. any options provided in the setup key in the first optional argument of the \mgls-like command.

These options are described below.

7.9.1. Indexing

indexmain=⟨*value*⟩ *default:* **true**; *initial:* **true**

This setting may take one of the following values:

false don't index the main entry;

true index the main entry;

first only index the main entry if it's the first use (of the main entry).

indexothers=⟨*value*⟩ *default:* **true**; *initial:* **true**

This setting may take one of the following values:

false don't index the other entries;

true index the other entries;

first only index the other entries if it's the first use (of the non-main entry).

7.9.2. Location Formats (Encaps)

encapmain=⟨*value*⟩ *initial:* **glsnumberformat**

This setting value should be the value to pass to `format` option (location encap) for the main entry.

encapothers=⟨*value*⟩ *initial:* **glsnumberformat**

This setting value should be the value to pass to `format` option (location encap) for the other (not main) entries.

7.9.3. Post-Link Hooks

postlinks=⟨*value*⟩ *initial:* **none**

This setting determines whether or not to enable the individual element's post-link hook. The value may be one of:

none suppress the post-link hook for all elements;

all don't suppress the post-link hook for all elements;

notlast only suppress the post-link hook for the last element;

mainnotlast suppress the post-link hook for all "other" (not main) elements and for the last element (so only the main element will have its post-link hook as long as it's not the last element);

mainonly suppress the post-link hook for all "other" (not main) elements;

othernotlast suppress the post-link hook for the main element and for the last element (so only the other elements will have their post-link hook as long as the element isn't the last one);

otheronly suppress the post-link hook for the main element.

mpostlink=⟨*value*⟩ *default:* **true**; *initial:* **true**

This setting determines whether or not to enable the multi-entry post-link hook (see §7.6). The value may be one of:

false suppress the multi-entry post-link hook;

true enable the multi-entry post-link hook;

firstonly enable the multi-entry post-link hook only for the first use of the multi-entry;

usedonly enable the multi-entry post-link hook only for the subsequent use of the multi-entry.

mpostlinkelement=⟨*value*⟩ *initial:* **last**

This setting indicates which post-link hook should be used if the multi-entry post-link hook has been enabled. Allowed values:

last use `\mglslastelementpostlinkhook` (that is, use the post-link hook for the last element);

main use `\mglslastmainpostlinkhook` (that is, use the post-link hook for the main element);

custom use `\mglscustompostlinkhook`.

Some combinations may cause a repeated hook.

7.9.4. Prefixes and Suffixes

See §7.3 for more information on prefixes and suffixes. Note that the prefixes and suffixes are not affected by case-changing commands such as `\Mgls` or `\MGLS`. If you want a prefix to obey case-changing, use the `\mpgls`-like commands instead (see §7.11.4).

firstprefix=⟨*value*⟩

The prefix to use on first use of the multi-entry.

usedprefix=⟨*value*⟩

The prefix to use on subsequent use of the multi-entry.

firstsuffix=⟨*value*⟩

The suffix to use on first use of the multi-entry.

usedsuffix=⟨*value*⟩

The suffix to use on subsequent use of the multi-entry.

7.9.5. Skipping Elements

The skip options apply to the multi-entry first use flag not the individual element first use. See §7.7.

firstskipmain=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If `true`, the main element will be omitted on (multi-entry) first use.

firstskipothers=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If `true`, the other (non-main) elements will be omitted on (multi-entry) first use.

usedskipmain=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If `true`, the main element will be omitted on (multi-entry) subsequent use.

usedskipothers=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If `true`, the other (non-main) elements will be omitted on (multi-entry) subsequent use.

Note that it is technically possible to set the skip options so that both the main and the other elements are skipped. However, by default, this will generate a warning and only the final optional argument (the ⟨*insert*⟩) will be displayed. There won't be a loop over all elements so the commands set at each iteration, such as `\mglscurrentlabel`, won't be defined.

The warning and the insertion of the ⟨*insert*⟩ is done by:

`\glsxtrmglsWarnAllSkipped{`⟨*message*⟩`}{`⟨*insert*⟩`}{`⟨*fmt-cs*⟩`}`

where ⟨*message*⟩ is the warning message and ⟨*cs*⟩ is the control sequence that encapsulates the entire content (including hyperlink and the `textformat` setting, if enabled).

If, for some particular reason, you want this scenario, you can redefine this command to omit the warning.

7.9.6. General

hyper=⟨*value*⟩ *initial:* **individual**

This setting may take one of the following values:

none no hyperlinks;

allmain encapsulate the entire content with a single hyperlink to the main entry's target;

mainonly only hyperlink the main entry;

individual hyperlink each entry individually;

otheronly only hyperlink the other entries;

notmainfirst don't hyperlink the main entry on multi-entry first use;

nototherfirst don't hyperlink the other entries on multi-entry first use;

notfirst don't hyperlink any entries on multi-entry first use.

textformat=⟨*value*⟩ *initial:* **@firstofone**

This setting value should be the control sequence name (without the leading backslash) of the command used to encapsulate the entire content.

category=⟨*category-label*⟩

The category to apply to the multi-entry. This is independent of the categories of each of the elements. The default is no category. See §7.8.

mglsopts=⟨*option list*⟩

Default options to pass to commands like `\mgls`. Note that `setup` can't be used within this value.

7.10. `\mgls` Options

The ⟨*options*⟩ for `\mgls` (and similar commands) are listed below. Any additional options provided will be appended to the `all` value. For example:

```
\mgls[counter=chapter]{cbot}
```

is equivalent to:

```
\mgls[all=counter=chapter]{cbot}
```

Whereas:

```
\mgls[counter=chapter,all=counter=section]{cbot}
```

is treated as:

```
\mgls[all=counter=section,counter=chapter]{cbot}
```

which has the same effect as:

```
\mgls[all=counter=chapter]{cbot}
```

The descriptions below reference `\gls` which is used internally by `\mgls`. Replace this with `\glspl` etc as applicable for the variants, such as `\mglspl`.

setup=⟨*option list*⟩

The value should be a list of any options that can be passed to `\multiglossaryentrysetup`. These options will override any conflicting options that were supplied to `\multiglossaryentry` or `\multiglossaryentrysetup`. Note that `mglsopts` can't be used within this value.

all=⟨*option list*⟩

The value should be a list of any options that can be passed to `\gls`. These options will be passed to each instance of `\gls` and will override any conflicting setting in `setup`.

main=⟨*option list*⟩

The value should be a list of any options that can be passed to `\gls`. These options will be passed to the instance of `\gls` used for the main label and will override any conflicting setting in `all`.

others=⟨*option list*⟩

The value should be a list of any options that can be passed to `\gls`. These options will be passed to each instance of `\gls` used for the other (not main) labels and will override any conflicting setting in `all`.

hyper=⟨*boolean*⟩ *default:* **true**

A boolean option to determine whether or not to use hyperlinks (if supported). This may cause a conflict with other options, but is essentially provided to allow the starred version of `\mgls` to switch off all hyperlinks.

multiunset=⟨*value*⟩ *initial:* **global**

This only applies to the multi-entry first use flag, described in §7.7, not the first use flags of the elements. The value may be one of:

`global` globally unset the flag;

`local` locally unset the flag;

`none` don't unset the flag.

presetlocal=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option that governs whether or not the following options should have a local or global effect.

resetall=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to reset all elements *before* using `\gls`. This option refers to the individual entry's first use flag not the multi-entry first use flag. (This is similar to passing `prereset` to each `\gls` but it's also applied to any skipped elements.)

resetmain=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to reset the main entry's first use flag *before* using `\gls`.

resetothers=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to reset the first use flag of all the other (not main) elements *before* using `\gls`.

unsetall=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to unset all elements *before* using `\gls`. This option refers to the individual entry's first use flag not the multi-entry first use flag. (This is similar to passing `preunset` to each `\gls` but it's also applied to any skipped elements.)

unsetmain=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to unset the main entry's first use flag *before* using `\gls`.

unsetothers=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

A boolean option to determine whether or not to unset the first use flag of all the other (not main) elements *before* using `\gls`.

The `reset`... options all use:

```
\mglselementreset{⟨entry-label⟩}
```

The `unset`... options all use:

```
\mglselementunset{⟨entry-label⟩}
```

These take the `presetlocal` into account.

An alternative way of resetting the other elements is to use:

```
\mglsunsetothers{⟨multi-label⟩}
```

or for a local reset:

```
\mglslocalunsetothers{⟨multi-label⟩}
```

7.11. Variants of `\mgls`

The commands listed in this section all behave like `\mgls`. These (including `\mgls` itself) are collectively referred to as the `\mgls`-like commands. All commands unset the multi-entry first use flag (unless the `multiunset=none` option is applied). Only those commands that use the `\gls`-like commands (such as `\gls` or `\glspl`) will unset the individual entry's first use flag.

7.11.1. \gls-like

\mglspl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \glspl instead of \gls for each entry.

\mglsmainpl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \glspl instead of \gls for the main entry and \gls for all the other entries.

\Mgls[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \Gls for the first entry and \gls for the other entries.

\MGls[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \Gls for all entries.

\Mglspl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \Glspl for the first entry and \glspl for the remaining entries.

\Mglsmainpl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

The first entry uses the appropriate sentence case command. The plural form is used for the main entry. So, if the first entry is also the main entry, \Glspl is used, otherwise \Gls is used. For the remaining entries, \glspl will be used if the entry is the main one, otherwise \gls will be used.

\MGlspl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

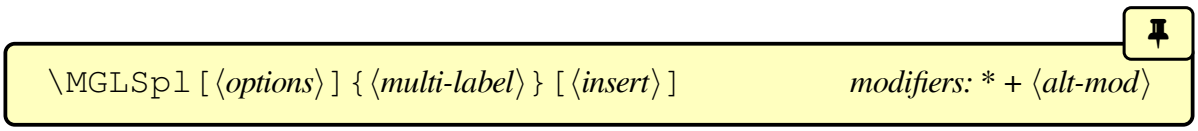
This uses \Glspl for all entries.

\MGlsmainpl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

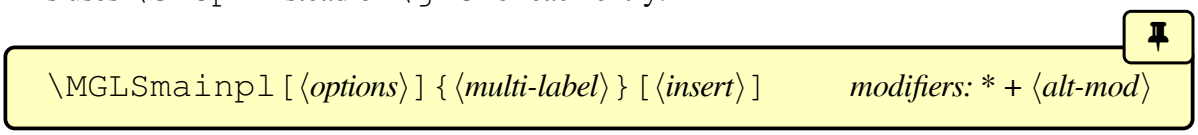
This uses \Glspl for the main entry and \Gls for the other entries.

\MGLS[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \GLS for all entries.

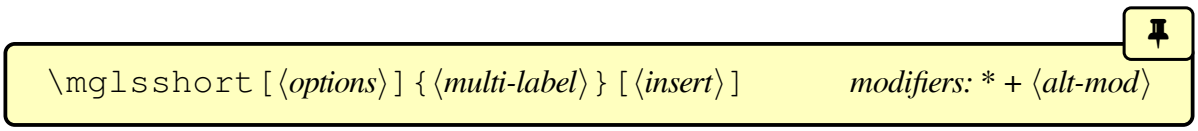


This uses \GLSpl instead of \gls for each entry.

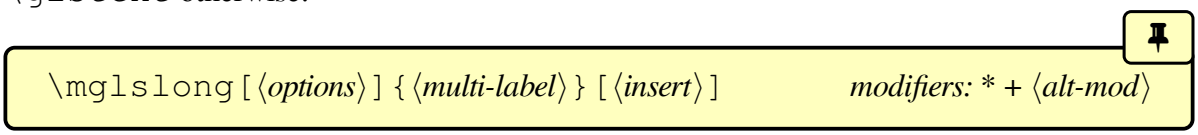


This uses \GLSpl for the main entry and \GLS for the others.

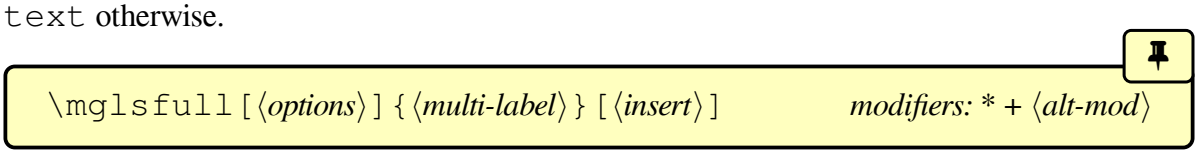
7.11.2. Abbreviations



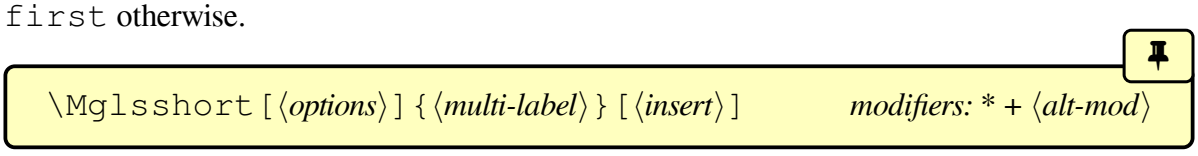
This will use \glsxtrshort for any entries that have the short field set and will use \glstext otherwise.



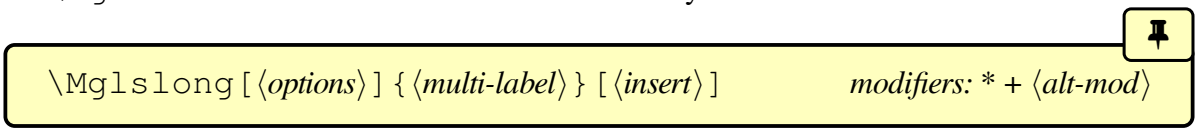
This will use \glsxtrlong for any entries that have the long field set and will use \glstext otherwise.



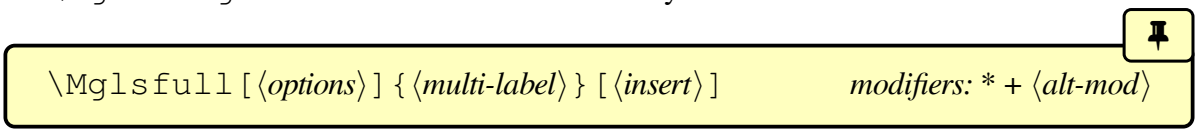
This will use \glsxtrfull for any entries that have the short field set and will use \glsfirst otherwise.



As \mglsshort but sentence case for the first entry.



As \mglslong but sentence case for the first entry.



As \mglsfull but sentence case for the first entry.

7.11.3. Other Fields

\mglsname[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \glsname for each entry.

\Mglsname[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \Glsname for the first entry and \glsname for the remaining entries.

\MGlsname[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \Glsname for each entry.

\mglssymbol[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \glssymbol for each entry if the symbol field has been set, otherwise it uses \gls.

\MGlssymbol[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

This uses \glssymbol if the symbol field has been set otherwise it uses \Gls for each element. (Note that no case change is applied to the symbol as this usually isn't appropriate.)

\Mglssymbol[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

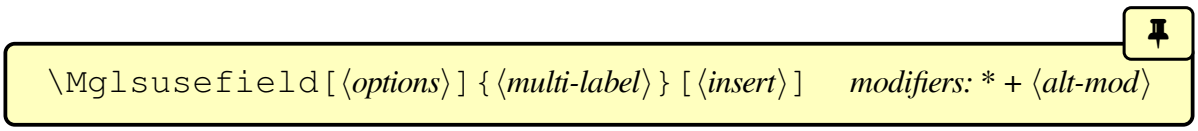
As \MGlssymbol, but \Gls is only used for the first element (if it doesn't have the symbol field set).

\mglsusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the field given by \mglsfield exists for an element, \glsdisp will be used for that element, with the link text obtained from the field value (followed by the ⟨*insert*⟩), otherwise \gls will be used.

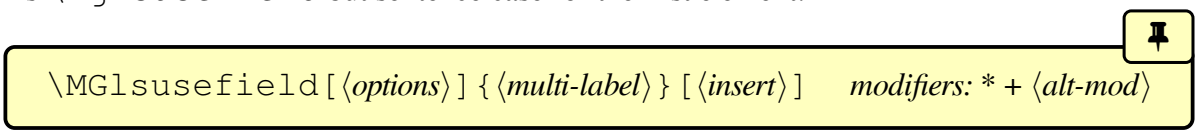
\mglsfield *initial:* useri

This command expands to the internal field label required by \mglsusefield. The default value is useri, which corresponds to the user1 key.



\Mglsusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \mglsusefield but sentence case for the first element.

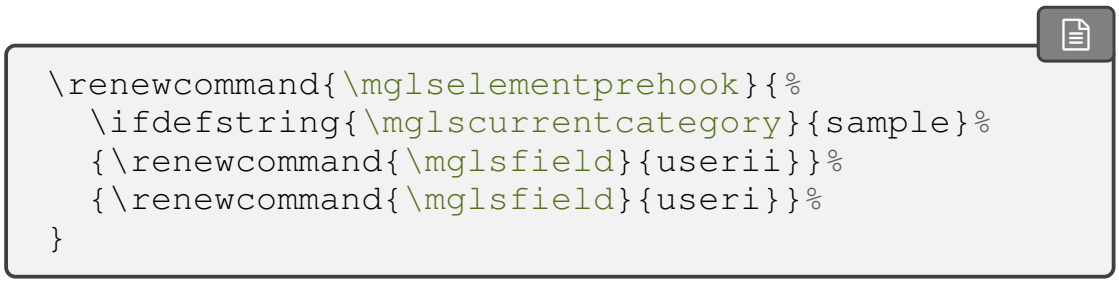


\MGlsusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \mglsusefield but sentence case for each element.

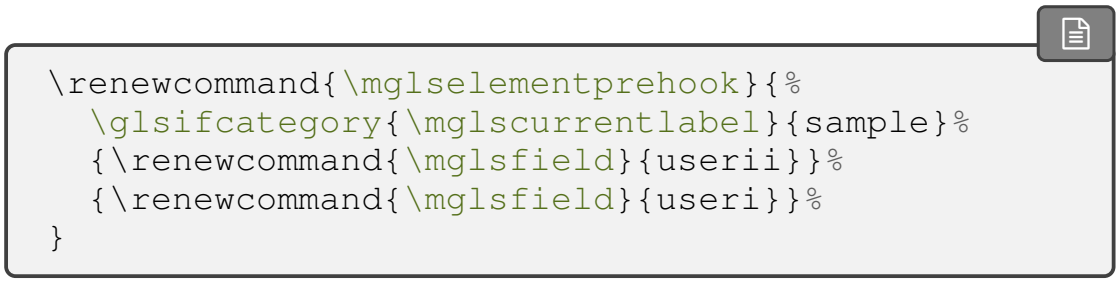
You can use the pre-element hook \mglselementprehook to locally redefine \mglsfield. Examples:

1. if the multi-category is "sample" then use the user2 field otherwise use the user1 field:



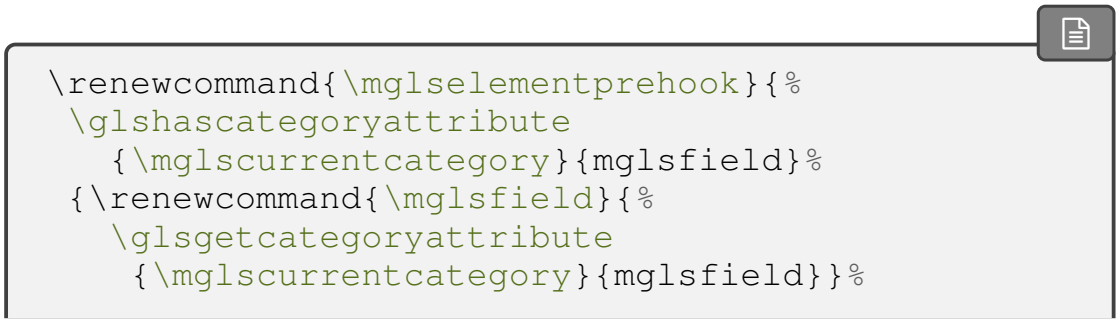
```
\renewcommand{\mglselementprehook}{%
  \ifdefstring{\mglscurrentcategory}{sample}%
  {\renewcommand{\mglsfield}{userii}}%
  {\renewcommand{\mglsfield}{useri}}%
}
```

2. if the element's category is "sample" then use the user2 field otherwise use the user1 field:



```
\renewcommand{\mglselementprehook}{%
  \glsifcategory{\mglscurrentlabel}{sample}%
  {\renewcommand{\mglsfield}{userii}}%
  {\renewcommand{\mglsfield}{useri}}%
}
```

3. if either the multi-entry's category or the element's category has the custom attribute "mglsfield" set then use it otherwise use the user1 field:



```
\renewcommand{\mglselementprehook}{%
 \glshascategoryattribute
   {\mglscurrentcategory}{mglsfield}%
 {\renewcommand{\mglsfield}{%
   \glsgetcategoryattribute
    {\mglscurrentcategory}{mglsfield}}%
```

```
  }%
  {%
    \glshasattribute
      {\mglscurrentlabel}{mglsfield}%
    {\renewcommand{\mglsfield}{%
      \glsgetattribute
        {\mglscurrentlabel}{mglsfield}}%
    }%
    {\renewcommand{\mglsfield}{useri}}%
  }%
}
```

7.11.4. Support for glossaries-prefix (\pgls)

If you load the glossaries-prefix package (either after glossaries-extra) or with the `prefix` package option, then the following commands will use one of the `\pgls`-like commands provided by that package. (See the glossaries user manual for further details.)

If the glossaries-prefix package hasn't been loaded then `\gls` (or analogous case-changing variant) will be used instead and a warning is issued with:

`\mpglsWarning`

This may be redefined to do nothing to remove the warning.

`\mpgls[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩]` *modifiers:* * + ⟨*alt-mod*⟩

Uses `\pgls` for the first element and `\gls` for the remaining elements.

`\mpglspl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩]` *modifiers:* * + ⟨*alt-mod*⟩

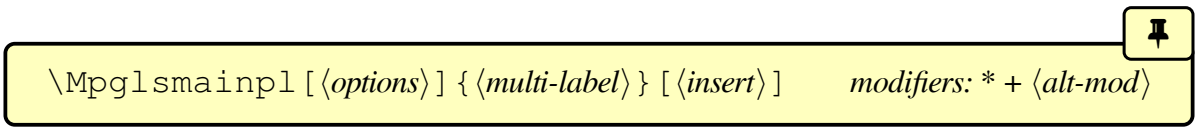
Uses `\pglspl` for the first element and `\glspl` for the remaining elements.

`\mpglsmainpl[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩]` *modifiers:* * + ⟨*alt-mod*⟩

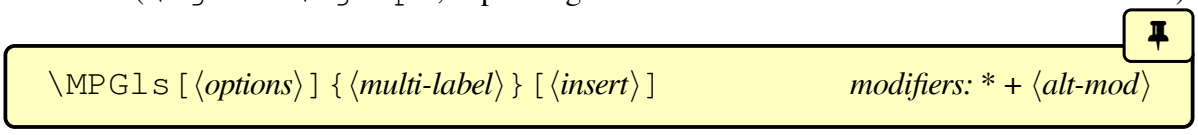
Only uses the plural form for the main element. The first element uses the prefixing command (`\pgls` or `\pglspl`, depending on whether the first element is the main element).

`\Mpgls[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩]` *modifiers:* * + ⟨*alt-mod*⟩

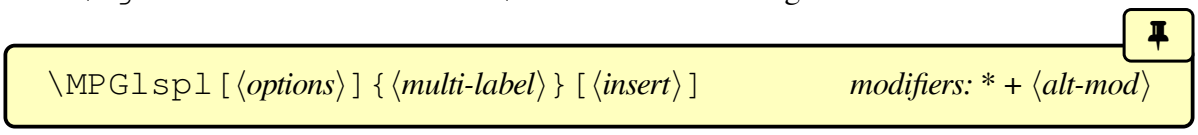
Uses `\Pgls` for the first element and `\gls` for the remaining elements.



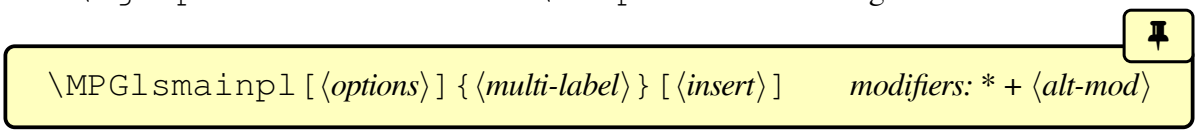
Only uses the plural form for the main element. The first element uses the sentence case prefixing command (`\Pgls` or `\Pglspl`, depending on whether the first element is the main element).



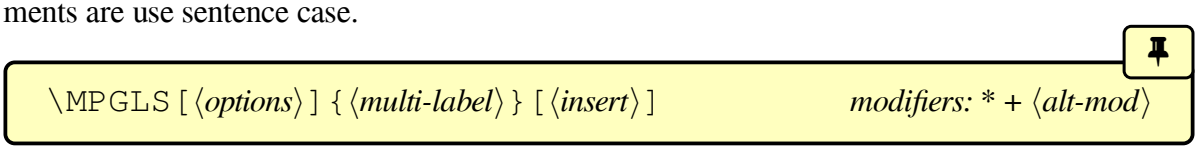
Uses `\Pgls` for the first element and `\Gls` for the remaining elements.



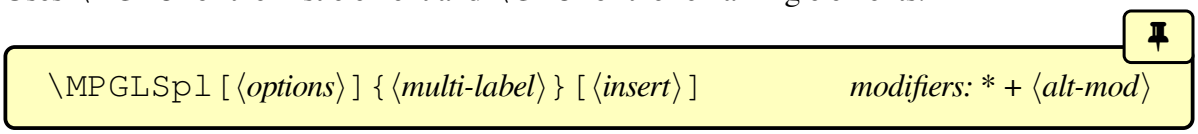
Uses `\Pglspl` for the first element and `\Glspl` for the remaining elements.



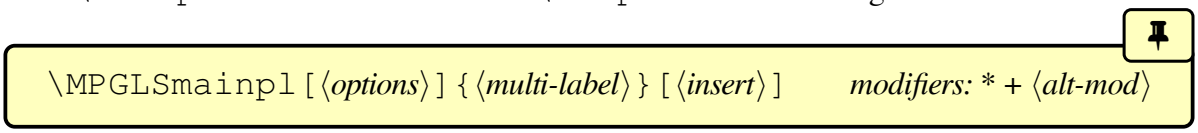
Only uses the plural form for the main element. The first element uses the prefixing command (`\Pgls` or `\Pglspl`, depending on whether the first element is the main element). All elements are use sentence case.



Uses `\PGLS` for the first element and `\GLS` for the remaining elements.



Uses `\PGLSpl` for the first element and `\GLSpl` for the remaining elements.



Only uses the plural form for the main element. All elements are converted to all caps. The first element uses the prefixing command (`\PGLS` or `\PGLSpl`, depending on whether the first element is the main element).

7.12. Cross-References

Multi-entry labels may be used in the cross-referencing keys `see` and `seealso`. The formatting command will use:

\mglsseefirstitem{⟨*multi-label*⟩}

for the first item in the list (if it's a multi-entry) and

\mglsseeitem{⟨*multi-label*⟩}

for any subsequent items that are multi-entries. The default definition of \mglsseeitem is:

```
\newcommand*{\mglsseeitem}[1]{%
 \mglsname[all=noindex,setup={hyper=allmain}]{#1}%
}
```

This switches off indexing, sets the hyperlink to encompass the entire multi-entry content and uses the name field. The default definition of \mglsseefirstitem is simply \mglsseeitem.

For example, to use the short or text fields:

```
\renewcommand*{\mglsseeitem}[1]{%
 \mglsshort[all=noindex,setup={hyper=allmain}]{#1}%
}
```

A multi-entry label may also be used in the alias key. The hyperlink target will be the target for the main entry. For example:

```
\multiglossaryentry{cbot}{clostridium,botulinum}
\newglossaryentry{botox}{name={botox},
 description={},alias={cbot}}
```

In this case \gls{botox} will hyperlink to the botulinum target.

Any multi-entries used in the cross-referencing keys must be defined before the glossary is displayed. There is some support for docdef=true but not for the other docdef settings.

7.13. Additional Commands

You can test if a label represents a multi-entry using:

```
\glsxtrifmulti{⟨multi-label⟩}{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if a multi-entry has been defined with the label ⟨*multi-label*⟩ otherwise it does ⟨*false*⟩.

```
\glsxtrmultimain{⟨multi-label⟩}
```

Expands to the main entry label for the multi-entry identified by ⟨*multi-label*⟩ or nothing if not defined.

```
\glsxtrmultilist{⟨multi-label⟩}
```

Expands to the list of element labels for the multi-entry identified by ⟨*multi-label*⟩ or nothing if not defined.

```
\mglsforelements{⟨multi-label⟩}{⟨cs⟩}{⟨body⟩}
```

Iterates over all the list of element labels for the multi-entry identified by ⟨*multi-label*⟩. This defines ⟨*cs*⟩ to the current element label on each iteration of the loop, which can be used to reference the label in ⟨*body*⟩. This internally uses `\@for` (patched by the xfor package, which allows the loop to be broken).

```
\mglsforotherelements{⟨multi-label⟩}{⟨cs⟩}{⟨body⟩}
```

As `\mglsforelements` but skips the main entry label.

```
\glsxtrmultitotalelements{⟨multi-label⟩}
```

Expands to the total number of elements in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined.

```
\glsxtrmultimainindex{⟨multi-label⟩}
```

Expands to the index of the main element in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined. Indexing starts from 1 for the first element.

```
\glsxtrmultilastotherindex{⟨multi-label⟩}
```

Expands to the index of the final non-main element in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined.

The \multiglossaryentry command will write the label information to the aux file using:

\writemultiglossentry{⟨options⟩}{⟨multi-label⟩}{⟨main-label⟩}{⟨list⟩}

This is will write the following line to the aux file:

\@glsxtr@multientry{⟨options⟩}{⟨multi-label⟩}{⟨main-label⟩}{⟨list⟩}

This is provided to support docdef=true and also for the benefit of any tools that require the information (such as bib2gls or autocompletion tools). If it's not required and causes too much clutter, it can be disabled by redefining \writemultiglossentry to do nothing.

7.14. bib2gls

In the bib2gls v2.9+ user manual, these multi-entry sets are referred to as "compound entries" or "compound sets" to differentiate them from bib2gls's multi-entry types (such as @dualentry).

Each instance of one of the \mgls-like commands is written to the aux file and so can be picked up by bib2gls (at least version 2.9). The resource option can be used to determine whether or not to consider the other (non-main) elements to be dependent on the main element.

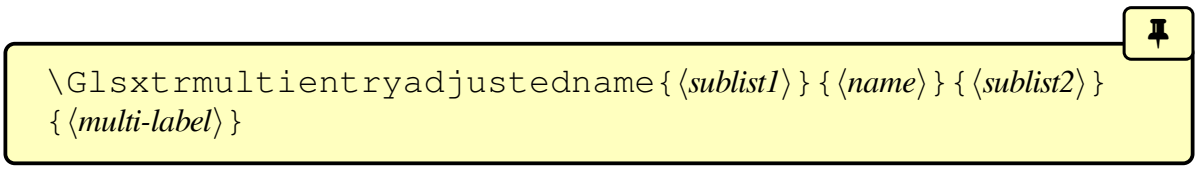
With bib2gls, you can either define compound entries in the document with \multiglossaryentry (or \providemultiglossaryentry) or you can define compound entries in the bib file with the @compoundset entry type. Whichever method you use, remember that the entries that form the elements of the set must be defined first. See the bib2gls manual (v2.9+) for further details.

You can use the resource option compound-adjust-name to replace the name field of the main entry to:

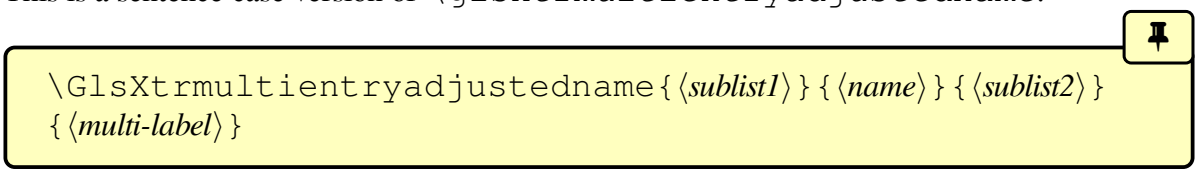
\glsxtrmultientryadjustedname{⟨sublist1⟩}{⟨name⟩}{⟨sublist2⟩}{⟨multi-label⟩}

where ⟨*multi-label*⟩ is the label identifying the compound set, ⟨*name*⟩ was the value of the name before adjustment, ⟨*sublist1*⟩ is the sub-list of other element labels before the main element (empty if the main element is the first element in the list), and ⟨*sublist2*⟩ is the sub-list of other elements after the main element (empty if the main label is the final element).

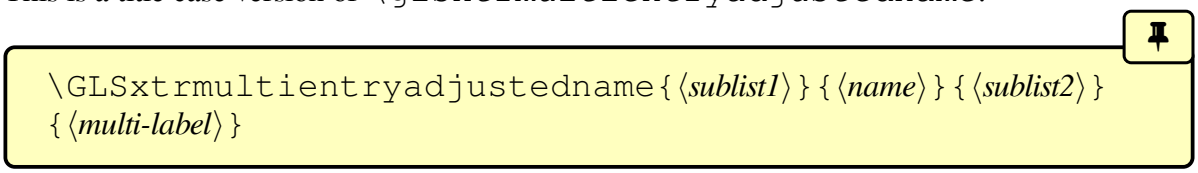
This command is defined in glossaries-extra-bib2gls, which is automatically loaded with record=only and record=nameref. Case-changing versions of this command are also available.



This is a sentence case version of \glsxtrmultientryadjustedname.



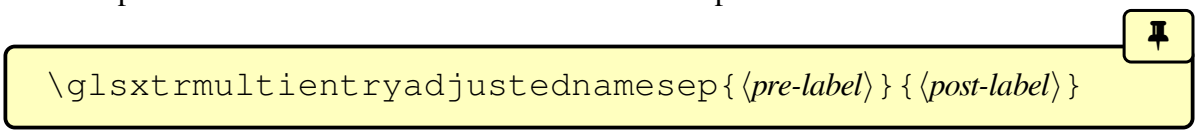
This is a title case version of \glsxtrmultientryadjustedname.



This is an all caps version of \glsxtrmultientryadjustedname.

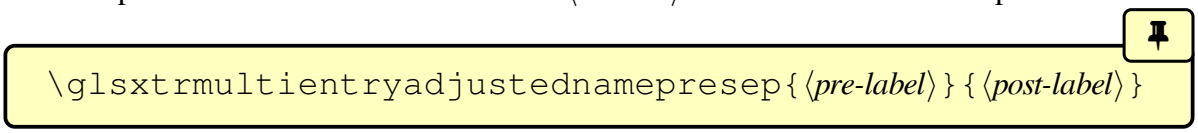
Note that the above commands don't take the prefix or suffix into account (see §7.3).

The separator between each element in the sub-lists is produced with:

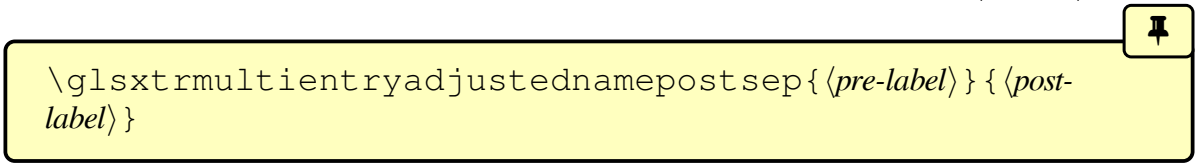


The default definition just uses \glscombinedfirstsepfirst.

The separator between the last element of ⟨*sublist1*⟩ and the main element is produced with:

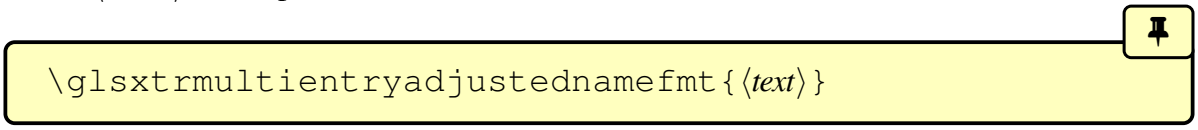


Similarly for the separator between the main element and the first element of ⟨*sublist2*⟩:

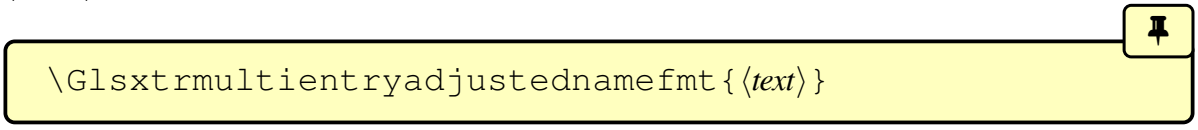


These both default to \glsxtrmultientryadjustednamesep.

The ⟨*name*⟩ is encapsulated with:



This just does its argument by default. If ⟨*sublist1*⟩ is empty for the sentence case version, then ⟨*name*⟩ is encapsulated with:



This does \makefirstuc{text} by default. For the title case version, the name is encapsulated with:

```
\GlsXtrmultientryadjustednamefmt{⟨text⟩}
```

This uses \glscapitalisewords, if defined, or \capitalisewords otherwise. The all caps version uses:

```
\GLSxtrmultientryadjustednamefmt{⟨text⟩}
```

This uses \mfirstucMakeUppercase by default.

Each element label in the sub-lists is encapsulated with:

```
\glsxtrmultientryadjustednameother{⟨text⟩}
```

This does \glsentryname{⟨label⟩} by default. For the sentence case version (where ⟨*sublist1*⟩ isn't empty), then the element label is encapsulated with:

```
\Glsxtrmultientryadjustednameother{⟨text⟩}
```

This does \Glsentryname{⟨label⟩} by default. The title case version uses:

```
\GlsXtrmultientryadjustednameother{⟨text⟩}
```

This does \glsentrytitlecase{⟨label⟩}{name} by default. The all caps version uses:

```
\GLSxtrmultientryadjustednameother{⟨text⟩}
```

This is defined as:

```
\newcommand*{\GLSxtrmultientryadjustednameother}[1]{%
 \mfirstucMakeUppercase{\glsentryname{#1}}}
```

8. Defining and Displaying Glossaries

As with the base glossaries package, you need to establish the indexing method in the preamble and use the appropriate \print⟨...⟩glossary command. For example, to use the "noidx" method you need \makenoidxglossaries in the preamble and \printnoidxglossary in the document. Whereas if you want to use makeindex or xindy, you need \makeglossaries in the preamble and \printglossary in the document. The glossaries-extra package provides a hybrid approach:

\makeglossaries[⟨*types*⟩]

If the optional argument is present, then ⟨*types*⟩ should be a comma-separated list of glossary labels that should be processed by makeindex or xindy as per the normal behaviour of \makeglossaries. Any non-ignored glossaries that are not listed in ⟨*types*⟩ should be treated as though \makenoidxglossaries was used. This means that the glossaries listed in ⟨*types*⟩ should be displayed using \printglossary and the other (non-ignored glossaries) should be displayed with \printnoidxglossary. See the accompanying file sample-mixedsort.tex for an example.

The hybrid \makeglossaries[⟨*list*⟩] doesn't automatically implement sanitizesort=false, which is the default with \makenoidxglossaries. However, since the hybrid option is typically used to allow sorting by definition or by use this won't usually make a difference.

If the optional argument is omitted, it will be treated as per the original \makeglossaries provided by the base glossaries package. If the optional argument is present, \glsindexingsetting will be set to makeindex-noidx or xindy-noidx, depending on whether makeindex or xindy should be used. If the upmendex option was used, then makeindex will be replaced with upmendex.

Glossaries can't be defined after \makeglossaries. This ensures that all the required indexing files are opened. If you're not using \makeglossaries, glossaries need to be defined before any entries that should belong to them are defined.

The base glossaries package provides \newignoredglossary to define an ignored glossary that doesn't have any associated indexing files. This will automatically switch off hy-

perlinks for any entries assigned to the glossary (since there will be no target). With glossaries−extra, it's possible to have targets without using the indexing methods provided by the base package. For example, it's possible to have standalone entries (see §8.5) or targets can be created with \printunsrtglossary, so glossaries−extra provides a starred version.

\newignoredglossary*{⟨glossary-label⟩}

This behaves like the unstarred version but doesn't disable hyperlinks. The glossary will still be omitted by iterative commands, such as \printunsrtglossaries, and can't be used with \printglossary or \printnoidxglossary. If you use an ignored glossary with \printunsrtglossary, you will need to use the title option to override the default title, if required.

\provideignoredglossary{⟨glossary-label⟩} *modifier: **

This is like \newignoredglossary but does nothing if the glossary has already been defined. The unstarred version disables hyperlinks.

With the indexing options provided by the base glossaries package, if you want a term to appear in more than one glossary, it's necessary to define a duplicate entry with a different label. With the "unsrt" family of commands, the same entry can appear in multiple glossaries. This can be done by simply copying the entry's label to the required glossary using:

\glsxtrcopytoglossary{⟨entry-label⟩}{⟨glossary-type⟩} *modifier: **

This just adds the label to the target glossary's internal comma-separated list that commands like \printunsrtglossary iterate over. The unstarred version locally adds the label. The starred version performs a global change.

\glsxtrcopytoglossary is not compatible with docdef=true.

Note that the type field will still be set to the original glossary. This is considered the entry's primary glossary. There's no field that keeps track of the additional glossaries the entry has been copied to.

If used with \printglossary or \printnoidxglossary, the entry will only be indexed for its primary glossary. It won't show up in the other glossaries, but will be found when using an iterative command, such as \glsaddall, over the target glossary.

You can test if an entry has already been added to a glossary with:

\GlsXtrIfInGlossary{⟨entry-label⟩}{⟨glossary-type⟩}{⟨true⟩}{⟨false⟩}

This does ⟨*true*⟩ if the entry given by ⟨*entry-label*⟩ is in the internal list of the glossary identified by ⟨*glossary-type*⟩, otherwise it does ⟨*false*⟩. If the glossary doesn't exist, this does ⟨*false*⟩ and will either generate an error (`undefaction=error`) or a warning (`undefaction=warn`). This command considers ignored glossaries as existing.

You can test if a glossary is empty with:

```
\glsxtrifemptyglossary{⟨glossary-type⟩}{⟨true⟩}{⟨false⟩}
```

This does ⟨*true*⟩ if the glossary identified by ⟨*glossary-type*⟩ is empty, otherwise does ⟨*false*⟩. If the glossary doesn't exist, this does ⟨*true*⟩ and will either generate an error (`undefaction=error`) or a warning (`undefaction=warn`). This command considers ignored glossaries as existing.

To test for the existence of a glossary, you can use `\ifglossaryexists` and `\doifglossarynoexistsordo`, which are documented in the "Conditionals" section of the glossaries user manual.

The base glossaries package provides `\forallglossaries` to iterate over a list of glossaries labels (all non-ignored glossaries by default). This can also be used with glossaries−extra but `\forallacronyms` is only for glossaries that have been declared as lists of acronyms, so it's inappropriate with the glossaries−extra package. Instead, you can use the analogous command:

```
\forallabbreviationlists{⟨cs⟩}{⟨body⟩}
```

Each instance of `\newabbreviation` will add the abbreviation's associated glossary (identified by the `type` key) to the internal list of labels (if not already added). Note that this won't take into account any glossaries that had abbreviations copied or moved to it.

8.1. Entry Page Reference

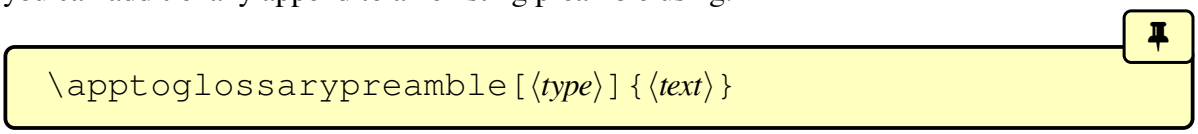
The base glossaries package provides `\glsrefentry`, which uses `\ref` to reference the entry's associated counter (enabled with `entrycounter` or `subentrycounter`, not the location counter). The glossaries−extra package additionally provides:

```
\glsxtrpageref{⟨entry-label⟩}
```

This works in the same way as `\glsrefentry` but uses `\pageref` instead of `\ref`. As with `\glsrefentry`, if the corresponding counter has not been enabled, `\glsxtrpageref` just does `\gls{⟨entry-label⟩}`.

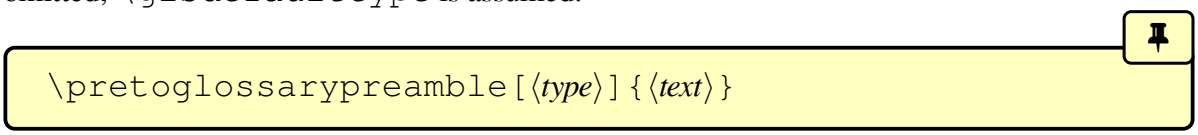
8.2. Glossary Preamble

The base package provides `\glossarypreamble`, which is used at the start of the glossary. By default, this will use the preamble associated with the current glossary. If you redefine `\glossarypreamble`, this will set the preamble for all glossaries. To set the preamble for a particular glossary, you can use `\setglossarypreamble`. With glossaries-extra, you can additionally append to an existing preamble using:



`\apptoglossarypreamble[`⟨*type*⟩`]{`⟨*text*⟩`}`

This (locally) appends ⟨*text*⟩ to the preamble for the glossary identified by ⟨*type*⟩. If ⟨*type*⟩ is omitted, `\glsdefaulttype` is assumed.

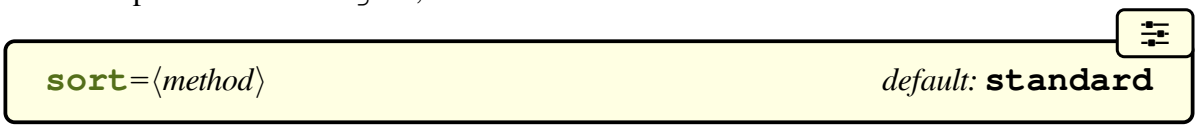


`\pretoglossarypreamble[`⟨*type*⟩`]{`⟨*text*⟩`}`

This (locally) prepends ⟨*text*⟩ to the preamble for the glossary identified by ⟨*type*⟩. If ⟨*type*⟩ is omitted, `\glsdefaulttype` is assumed.

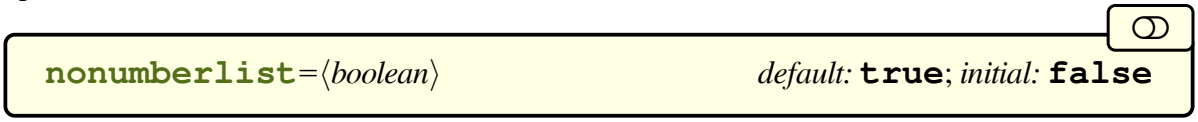
8.3. Options

The base glossaries package provides options that may be used with `\printglossary` and `\printnoidxglossary`. The glossaries-extra package provides additional options, some of which are specific to `\printunsrtglossary` and printunsrtglossarywrap. These extra options are listed below. Additionally, options provided by the base package that aren't available with one or more of the "unsrt" family of commands, or that may be replaced by a resource option with `bib2gls`, are also listed.



`sort`=⟨*method*⟩ *default:* **`standard`**

The `sort` option is provided by the base glossaries package, but is only available for `\printnoidxglossary`. Available methods are described in the glossaries user manual. The `\printunsrtglossary` and `\printunsrtinnerglossary` commands simply iterate over the glossary's internal list in the order in which the entries have been added to that glossary, so this option is not applicable. If you are using `bib2gls`, use the `sort` resource option instead.



`nonumberlist`=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

The `nonumberlist` option is provided by the base glossaries package. It may be used with `\printunsrtglossary` and `\printunsrtinnerglossary` but if you are using `bib2gls`, you may prefer to use the `save-locations=false` resource option instead, if location lists are not required. Note that `nonumberlist=false` will have no

effect with the save-locations=false resource option as there won't be any location lists to display.

title=⟨*text*⟩

The title option is provided by the base glossaries package to override the default title for the glossary. This option is also available for \printunsrtglossary and printunsrtglossarywrap but not for \printunsrtinnerglossary.

toctitle=⟨*text*⟩

The toctitle option is provided by the base glossaries package to override the default table of contents title for the glossary. This option is also available for \printunsrtglossary and printunsrtglossarywrap but not for \printunsrtinnerglossary.

numberedsection=⟨*value*⟩ *default:* **nolabel**; *initial:* **false**

The numberedsection option is provided by the base glossaries package to indicate whether or not the section header used at the start of the glossary should be numbered rather than unnumbered (and whether or not to automatically label the glossary with \label{\glsautoprefix⟨*glossary-type*⟩}). The numberedsection package option will change the default setting to match. This option is not available for \printunsrtinnerglossary.

style=⟨*style-name*⟩

The style option is provided by the base glossaries package to set the glossary style. This option is not available for \printunsrtinnerglossary.

label=⟨*label*⟩

The label option is provided by glossaries-extra to add \label{⟨*label*⟩} after the section header. This is essentially like numberedsection=nameref but you supply the label. This option is not available for \printunsrtinnerglossary. Alternatively, you can use:

\glsxtrsetglossarylabel{⟨*label*⟩}

This will need to be scoped or changed between glossaries or use a command in ⟨*label*⟩ that expands differently for each glossary to avoid duplicate labels.

If the supplied value is empty, the label is suppressed (without otherwise altering the numberedsection setting).

leveloffset=⟨*offset*⟩ *initial:* **0**

The leveloffset option sets or increments the hierarchical level offset. If ⟨*offset*⟩ starts with ++ then the current offset is incremented by the given amount otherwise the current offset is set to ⟨*offset*⟩. For example, an entry with a normal hierarchical level of 1 will be treated as though it has hierarchical level 1 + ⟨*offset*⟩. Note that the glossary style may not support the resulting hierarchical level. This option is only available for the "unsrt" family of commands and the printunsrtglossarywrap environment. See §8.4.3.1 for an example.

flatten=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

The flatten option treats all entries as though they have the same hierarchical level (which will be the value of leveloffset). This option is only available for the "unsrt" family of commands and the printunsrtglossarywrap environment. Unlike the flatten resource option, this option doesn't actually remove the parent field.

groups=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

The groups option is only applicable to the "unsrt" family of commands and printunsrtglossarywrap. If set to false, it will prevent groups from being formed. If true (the default), groups will only be formed if they are supported. See §8.4 for further details.

preamble=⟨*text*⟩

The preamble option redefines \glossarypreamble to ⟨*text*⟩.

postamble=⟨*text*⟩

This postamble option redefines \glossarypostamble to ⟨*text*⟩.

prefix=⟨*prefix*⟩

The prefix option is provided by glossaries-extra and simply redefines \glolinkprefix to expand to ⟨*prefix*⟩. If hyperlinks are supported and the glossary style uses \glstarget to create the entry's hypertarget, the target name is obtained from \glolinkprefix ⟨*entry-label*⟩. If you are displaying multiple glossaries with shared entries (for example, using the secondary resource option with bib2gls), then changing the prefix can avoid

duplicate targets. Alternatively, you can redefine \glstarget to use \glsxtrtarget.

Note that this option will also affect the targets used by the \gls-like and \glstext-like commands. This means that if you have, for example, \gls in the description of an entry, then its hyperlink will go to that entry's item in the current glossary. Whereas referencing that entry outside of the glossary will hyperlink to the glossary that uses the prefix matching the setting at that point in the document.

Example 133 has two glossaries with different target prefixes:

133

```
\usepackage[colorlinks]{hyperref}
\usepackage[showtargets=annoteleft,style=tree]
{glossaries-extra}
\newglossaryentry{sample}{name={sample},
 description={an example description that references
  \gls{another}}}
\newglossaryentry{another}{name={another},
 description={some other example description
  that references \gls{sample}}}
\begin{document}
Link to glossary 1: \gls{sample}.

Link to glossary 2: \gls[prefix=other-]{sample}.
\printunsrtglossary
\printunsrtglossary[prefix=other-]
\end{document}
```

This uses the showtargets package option to show the target names to the left of the hyperlink or hypertarget.

Within the main part of the document, the first reference to "sample" has a hyperlink to the first glossary (with the target glo:sample, which uses the default prefix), and the second reference has a hyperlink to the second glossary (with the target other-sample).

Within the glossaries, the \gls references use the current glossary prefix, so the target is in the same glossary.

targetnameprefix=⟨*prefix*⟩

The targetnameprefix option is similar to the prefix option but only affects the prefix for the entry item's target and doesn't change the prefix for any references contained within the glossary. This *prepends* the given prefix to the default prefix.

Example 134 modifies the above example:

134

Example 133: Changing the target prefix

Link to glossary 1: [glo:sample]▹sample◃.
Link to glossary 2: [other-sample]▹sample◃.

Glossary

[glo:sample]▹**sample**◃ an example description that references [glo:another]▹another◃

[glo:another]▹**another**◃ some other example description that references [glo:sample]▹sample◃

Glossary

[other-sample]▹**sample**◃ an example description that references [other-another]▹another◃

[other-another]▹**another**◃ some other example description that references [other-sample]▹sample◃

```
Link to glossary 1: \gls{sample}.

Link to glossary 2: \gls[prefix=other-glo:]{sample}.
\printunsrtglossary
\printunsrtglossary[targetnameprefix=other-]
```

Example 134: Prepending to the target prefix for just the entry item

Link to glossary 1: [glo:sample]▹sample◃.
Link to glossary 2: [other-glo:sample]▹sample◃.

Glossary

[glo:sample]▹**sample**◃ an example description that references [glo:another]▹another◃

[glo:another]▹**another**◃ some other example description that references [glo:sample]▹sample◃

Glossary

[other-glo:sample]▹**sample**◃ an example description that references [glo:another]▹another◃

[other-glo:another]▹**another**◃ some other example description that references [glo:sample]▹sample◃

Note that this has prepended `other-` to the existing `glo:` prefix. This is why the `prefix` option in the second `\gls` reference had to be changed to match the appropriate hypertarget name. The `\gls` references in the second glossary now point to the relevant line in the first glossary.

It's possible to combine `targetnameprefix` with `prefix={}` but that will also affect the `\gls` references within the glossary.

target=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

The `target` option is a boolean option that can be used to switch off the automatic creation of the entry hypertargets but still allows hyperlinks within the glossary. This can be used to prevent duplicate destinations for secondary glossaries.

8.4. Displaying a Glossary Without Sorting or Indexing

The base glossaries package provides two ways of displaying a glossary, depending on the indexing option: `\printglossary` (which inputs a file created by `makeindex` or `xindy` that contains the code to typeset the glossary) or `\printnoidxglossary` (which doesn't require an external indexing application but instead uses LaTeX to sort a list of entry labels and then iterates over that list, where the location list information is picked up from the `aux` file).

The glossaries-extra package provides an alternative that doesn't require sorting or indexing.

`\printunsrtglossary[`⟨*options*⟩`]`

This behaves in a similar way to `\printnoidxglossary`, but it always lists all the defined entries for the given glossary in the order in which they were added to the glossary. Unlike `\printglossary`, you may use `\printunsrtglossary` with an ignored glossary.

The "unsrt" part of the command name indicates that the list is always in order of definition (unsorted). This command may be used with `bib2gls` which ensures that the order of definition matches the desired order as given by the `sort` resource option (and other applicable options). However, `\printunsrtglossary` may simply be used without `bib2gls`, in which case you need to ensure that you define your glossary entries in the required order.

The "unsrt" family of commands and printunsrtglossarywrap are not intended for use with `\makeglossaries` and `\makenoidxglossaries`. Mixing these different methods can result in unexpected behaviour.

There is also a starred version which has a mandatory argument:

`\printunsrtglossary*[`⟨*options*⟩`]{`⟨*init-code*⟩`}`

This is equivalent to:

```
\begingroup
  ⟨init-code⟩\printunsrtglossary[⟨options⟩]
\endgroup
```

There's no significant difference between doing:

```
{⟨init-code⟩\printunsrtglossary[⟨options⟩]}
```

and

```
\printunsrtglossary*[⟨options⟩]{⟨init-code⟩}
```

Note that unlike \glossarypreamble, the supplied ⟨*init-code*⟩ is done before the glossary header.

If you want to use one of the tabular-like styles with \printunsrtglossary, make sure you load glossaries-extra-stylemods which modifies the definition of \glsgroupskip to avoid the "Incomplete \iftrue" error that may otherwise occur.

As with \printglossary and \printnoidxglossary, there is also a command to print all non-ignored glossaries in the order in which they were defined:

```
\printunsrtglossaries
```

This means you now have the option to simply list all entries on the first LaTeX run without the need for a post-processor, however there will be no location list in this case, as that has to be set by a post-processor such as bib2gls (see §11).

No attempt is made to gather hierarchical elements. If child entries aren't defined immediately after their parent entry, they won't be together in the glossary when using \printunsrtglossary.

The way that \printunsrtglossary basically works is to iterate over every label in the glossary's internal label list and format each entry according to the way the glossary style would normally format the entry's hierarchical level (described in more detail in §8.4.3). If a change in letter group is detected, the letter group heading and group skip will be inserted.

A label is appended to the glossary's internal label list whenever an entry is defined. This means that the list will normally be in order of definition, but it's also possible to copy an entry's label to another glossary's internal label list using \glsxtrcopytoglossary, which can be used to provide a different order.

Example 135 illustrates this method:

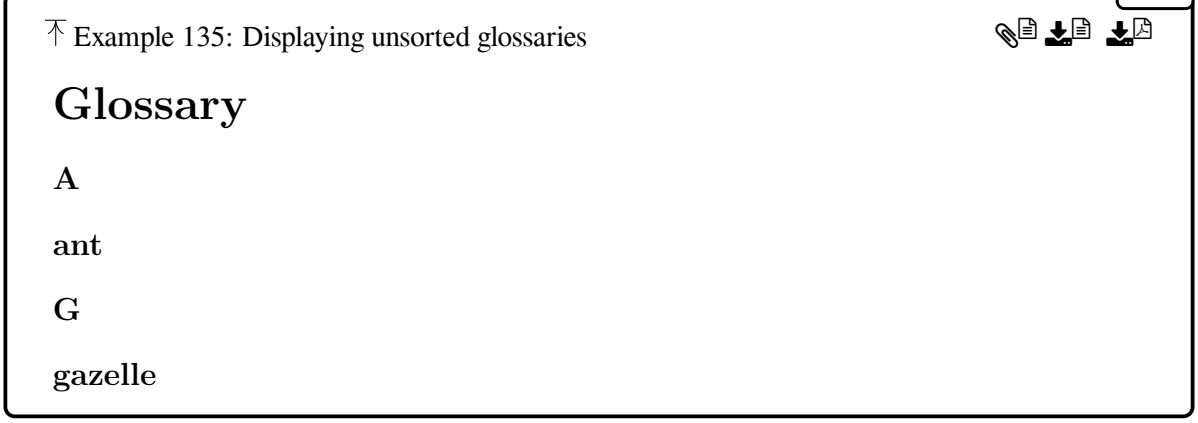
135

```
\documentclass{article}
\usepackage[style=treegroup]{glossaries-extra}
\newglossaryentry{ant}{name={ant},
 description={}}
\newglossaryentry{gazelle}{name={gazelle},
 description={}}
\begin{document}
\printunsrtglossary
```

```
\end{document}
```

The document build only requires one LaTeX call in this case.

Example 135: Displaying unsorted glossaries

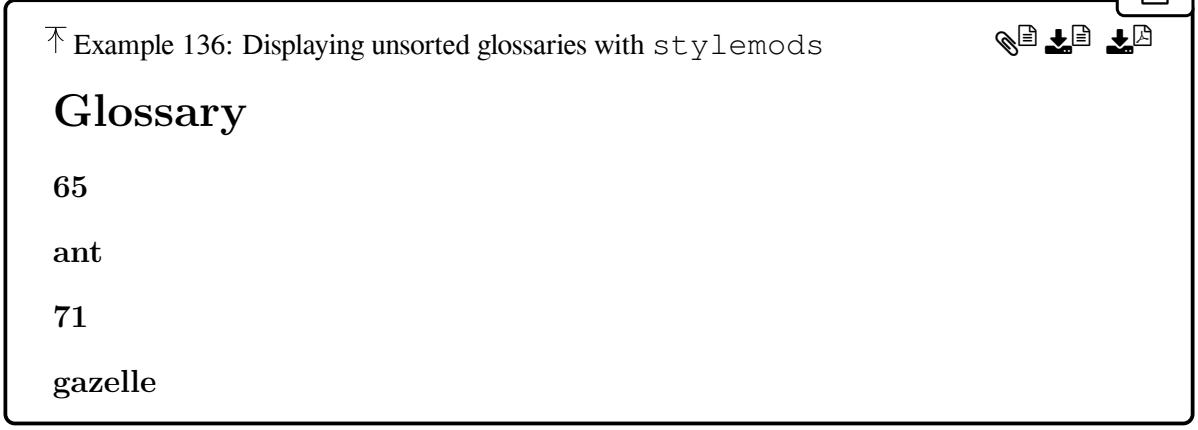


Be careful if you want letter groups with `\printunsrtglossary`.

Example 136 demonstrates the difference if the `stylemods` option is added: 136

```
\usepackage[stylemods,style=treegroup]{glossaries-
extra}
```

Example 136: Displaying unsorted glossaries with `stylemods`



In this case, the group headings are now numbers instead of letters. The styles provided with glossaries−extra and those modified by glossaries−extra−stylemods are designed to assist integration with `bib2gls`. Without these modifications, `\printunsrtglossary` behaves like the less sophisticated `\printnoidxglossary` which checks if the label is an integer less than 256 and uses `\char` to create the title (if no title has been provided).

If you really want to use `\printunsrtglossary` without `bib2gls` and you want letter groups with `stylemods` without having to define all the titles, you can use:

```
\glsxtrnoidxgroups
```

which will switch over to using the group titling method used with `\printnoidxglossary`. This command is only available with `record=off` and can't be used with `\makeglossaries`.

If you have at least v4.57 of the base glossaries package and at least v3.0 of the datatool package, `\printnoidxglossary` now obtains the letter group information from the datatool sorting function and sets it in a special internal field with the label `dtlsortgroup`. So if you use `\glsxtrnoidxgroups`, it will first test if that field has been set before falling back on its old behaviour.

If, conversely, you don't want any groups formed, regardless of the glossary style, you can disable them with `groups=false`.

Example 137 modifies Example 136 so that the document contains:

137

```
\printunsrtglossary[title={Glossary 1}]
\glsxtrnoidxgroups
\printunsrtglossary[title={Glossary 2}]
\printunsrtglossary[groups=false,
 title={Glossary 3}]
\printunsrtglossary[style=tree,nogroupskip,
 title={Glossary 4}]
```

This repeats the same glossary. The first is the same as Example 136. The second is the same as Example 135 (which didn't use `stylemods`). The final two glossaries have the groups suppressed. Using `groups=false` (Glossary 3) is more efficient than using `nogroupskip` and switching to a style that doesn't show the header (Glossary 4).

I've also switched to two column mode to display the result in a more compact form. The first two glossaries are shown on the left and the last two are on the right:

Example 137: Displaying unsorted glossaries with different group settings

Glossary 1

65

ant

71

gazelle

Glossary 2

A

ant

G

gazelle

Glossary 3

ant
gazelle

Glossary 4

ant
gazelle

The "unsrt" family of commands were designed for use with `bib2gls`, which uses more complex alphanumeric group labels to allow for greater customization and to avoid conflict where there are multiple glossaries or hierarchical levels with potentially the same letter groups.

The way that `bib2gls` works is to select entries from a `bib` file, according to the document requirements, sort the entries, and then write the entry definitions (with commands like `\longnewglossaryentry*` or `\newabbreviation`) in the `glstex` in the desired order, which is then input by `\GlsXtrLoadResources`. This means that `\printunsrtglossary` will display the entries in that order since, from glossaries−extra's point of view, that's the order of definition.

While it is possible to use `\printunsrtglossary` without `bib2gls`, as in the above example, for long or complex glossaries it's better to use `bib2gls` which can automatically assign appropriate titles to the groups.

Groups and hierarchy are discussed in more detail in §8.4.1. See §8.4.2 for location lists and §8.4.3.1. Advanced commands and further detail about the way `\printunsrtglossary` works are covered in §8.4.3.

8.4.1. Groups and Hierarchy

See Gallery: Logical Glossary Divisions (type vs group vs parent)[a] for the difference between the group, type and parent fields.

[a] dickimaw-books.com/gallery/index.php?label=logicaldivisions

Example 138 is a longer example that uses stylemods to automatically load glossary-bookindex: 138

```
\documentclass{article}
\usepackage[stylemods=bookindex,style=bookindex]
{glossaries-extra}
\newglossaryentry{waterfowl}{name={waterfowl},
 description={}}
\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{adder}{name={adder},
 description={}}
\newglossaryentry{duck}{name={duck},
 parent={waterfowl},
 description={}}
\newglossaryentry{zebra}{name={zebra},
 description={}}
\newglossaryentry{aardvark}{name={aardvark},
 description={}}
\newglossaryentry{gazelle}{name={gazelle},
 description={}}
\newglossaryentry{mallard}{name={mallard},
 parent={duck},
 description={}}

\newglossary*{another}{Another Glossary}
\glsxtrcopytoglossary{mallard}{another}
\glsxtrcopytoglossary{aardvark}{another}
\glsxtrcopytoglossary{zebra}{another}
\glsxtrcopytoglossary{ant}{another}
\glsxtrcopytoglossary{duck}{another}
\begin{document}
\printunsrtglossary
\printunsrtglossary[type=another]
\end{document}
```

Unlike the previous examples that defined the entries in alphabetical order, this example hasn't used any logical order. Note, in particular, that the child entries "duck" and "mallard" (which have the `parent` key set) have not been defined immediately after their parent.

The first `\printunsrtglossary` has the default `type=main` and lists all entries defined in the `main` glossary, in the order in which they were defined. The second `\printunsrtglossary` lists all entries in the custom `another` glossary and is in the order in which the entries were copied to that glossary.

The document build for Example 138 again simply requires one LaTeX call.

Example 138: Displaying unsorted glossaries with a copied list

Glossary

87

waterfowl

65

ant
adder
duck

90

zebra

65

aardvark

71

gazelle
mallard

Another Glossary

mallard

65

aardvark

90

zebra

65

ant
duck

There are some oddities in both lists. It's the glossary style that determines the formatting of the entries according to the entry's hierarchical level, but it looks strange for the duck and mallard entries to be indented when they don't follow after their parent entry.

As the internal loop within `\printunsrtglossary` iterates over each entry, it tries to determine which letter group the entry belongs to. If it's different from the group for the previous entry (in the same hierarchical level), a group header is added (which may or may not

be displayed, depending on the glossary style). This means than an unordered list of entries, such as in the above example, may contain repeated headers.

The way that the group is determined depends on whether or not the group key has been defined. If it isn't defined (the default), then the group label is obtained from the uppercase character code of the first token of the sort key. If the token doesn't have an uppercase character code (indicating that it's not a letter) or if the sort value is empty then the label will be set to glssymbols (which corresponds to the symbol group). This is the same way that \printnoidxglossary inserts groups.

Remember that if the sort key hasn't been set, it will be assigned automatically to the same value as the name key (or with sort=use or sort=def to a numerical value). The sort key will be empty if you use sort=clear. The sort=none setting simply skips the pre-processing of the sort key (such as sanitizing).

For example, the ant entry doesn't explicitly use the sort key, so the sort value is obtained from the name key, which is set to ant. The first token is "a", which is a letter. The group's label is obtained from the letter's uppercase character decimal code (65). There's no associated title (which can be assigned with \glsxtrsetgrouptitle), so the title is simply "65" (with stylemods, see earlier) or "A" (without stylemods or with \glsxtrnoidx-groups).

The ant entry is followed by "adder". The same process determines that the "adder" group label is also 65. There's no change in the group label from the previous entry (ant) so no header is inserted.

By default, this group check is omitted for child entries, which is why no group header is inserted before duck or mallard. So the next entry to be checked for a group is the zebra entry, which has the group label 90 (the decimal code for "Z"). Again there's no title associated with that label so the title is simply the label.

The zebra entry is followed by aardvark which, following the same process, has the group label 65. This is different from the previous group label (90) so a group header is inserted. This is why there are two "90" letter groups.

The "unsrt" family of commands don't order the entries.

If the group key has been defined (which is the case with record=only and record=nameref) then the group *label* is obtained from the group field. If the group field is defined but empty then the entry will belong to the empty group. The value of the sort field is now irrelevant.

So, simply adding the record option to the above example document will cause the group headers to disappear. This is because the group key will now be defined but is empty for each entry. Even with a style like bookindex, there won't be any group headers.

Provided the group key has been defined, the field used to store the group label is given by:

\glsxtrgroupfield *initial:* group

This expands to group, by default. However it's possible to use a different field in which to store the group label, in which case \glsxtrgroupfield will need to be redefined. For example:

```
{\renewcommand{\glsxtrgroupfield}{othergroup}%
\printunsrtglossary}
```

or

```
\printunsrtglossary*{%
 \renewcommand{\glsxtrgroupfield}{othergroup}}
```

(but this still requires the group key to be defined, even if it's not being used to store the group label). With bib2gls, the secondary resource option (combined with --group) will store the group label obtained from the secondary sort in the secondarygroup field and adds the redefinition of \glsxtrgroupfield to the associated glossary preamble. This prevents it from clashing with the group field in the event that the secondary sort method has produced a different set of groups (which is likely).

Example 139 uses record to create the group key and assigns group labels with associated titles. Note that the secondarygroup field doesn't have an associated key, so it needs to be set with a field assignment command, such as \GlsXtrSetField. 139

```
\documentclass{article}
\usepackage[record,stylemods=bookindex,
 style=bookindex]{glossaries-extra}

\glsxtrsetgrouptitle{group1label}{Group 1}
\glsxtrsetgrouptitle{group2label}{Group 2}
\glsxtrsetgrouptitle{group3label}{Group 3}
\glsxtrsetgrouptitle{group4label}{Group 4}


\newglossaryentry{waterfowl}{name={waterfowl},
 description={},
 group={group1label}}
\newglossaryentry{ant}{name={ant},
 description={},
 group={group1label}}
\GlsXtrSetField{ant}{secondarygroup}{group4label}
\newglossaryentry{adder}{name={adder},
```

```
 description={},
 group={group2label}}
\newglossaryentry{duck}{name={duck},
 parent={waterfowl},
description={},group={group4label}}
\GlsXtrSetField{duck}{secondarygroup}{group2label}
\newglossaryentry{zebra}{name={zebra},
 description={},
 group={group2label}}
\GlsXtrSetField{zebra}{secondarygroup}{group3label}
\newglossaryentry{aardvark}{name={aardvark},
 description={},
 group={group2label}}
\GlsXtrSetField{aardvark}{secondarygroup}
{group1label}
\newglossaryentry{gazelle}{name={gazelle},
 description={},
 group={group1label}}
\newglossaryentry{mallard}{name={mallard},
 parent={duck},description={},
 group={group2label}}
\GlsXtrSetField{mallard}{secondarygroup}
{group3label}


\newglossary*{another}{Another Glossary}
\glsxtrcopytoglossary{mallard}{another}
\glsxtrcopytoglossary{aardvark}{another}
\glsxtrcopytoglossary{zebra}{another}
\glsxtrcopytoglossary{ant}{another}
\glsxtrcopytoglossary{duck}{another}
\setglossarypreamble[another]
 {\renewcommand{\glsxtrgroupfield}{secondarygroup}}
\begin{document}
\printunsrtglossary
\printunsrtglossary[type=another]
\end{document}
```

This is essentially mimicking the way that the `secondary` resource option sets the `secondarygroup` field and adds the redefinition of `\glsxtrgroupfield` to the secondary glossary's preamble. (Although in this case, there's no logical order.)

Note that even though the duck and mallard entries have the `group` and `secondarygroup` fields set, there's no group title for them in either glossary because they are child entries.

Example 139: Displaying unsorted glossaries with custom groups

Glossary

Group 1

waterfowl
ant

Group 2

adder
    duck

zebra
aardvark

Group 1

gazelle
    mallard

Another Glossary

    mallard

Group 1

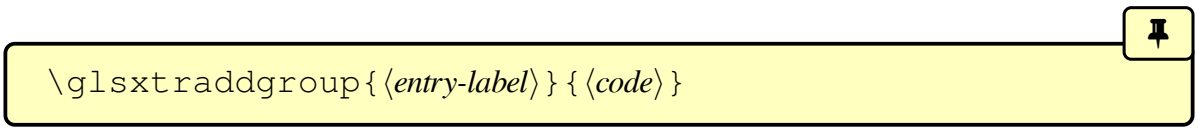
aardvark

Group 3

zebra

Group 4

ant
    duck

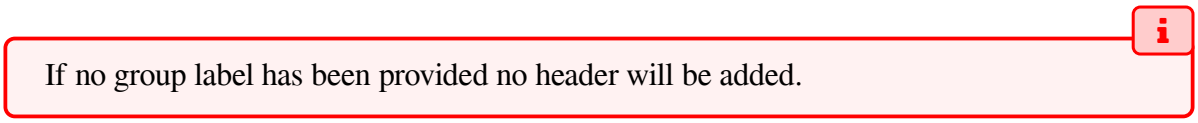


```
\glsxtraddgroup{⟨entry-label⟩}{⟨code⟩}
```

This command will perform ⟨*code*⟩ if the entry identified by ⟨*entry-label*⟩ should have group support (provided the `group` field has been set). The default definition is:

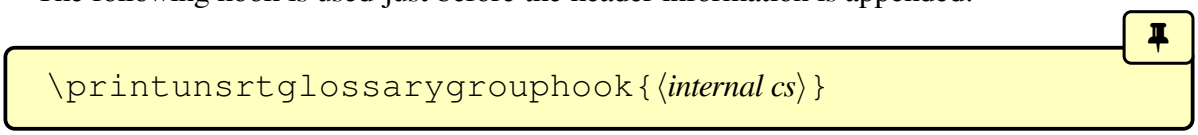
```
\newcommand*{\glsxtraddgroup}[2]{%
  \ifglsxtrprintglossflatten
    #2%
  \else
    \ifglshasparent{#1}{}{#2}%
  \fi
}
```

This means that only entries that don't have a parent (with `flatten=false`) or any entry (with `flatten=true`) will have the group check performed. With `bib2gls`, the `group-level` resource option will redefine `\glsxtraddgroup` to always do ⟨*code*⟩, which means that all entries will have the group check performed.



If no group label has been provided no header will be added.

The following hook is used just before the header information is appended:

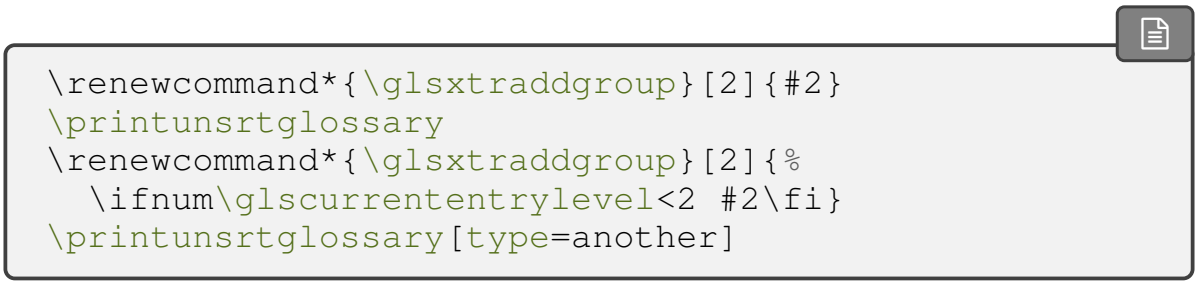


```
\printunsrtglossarygrouphook{⟨internal cs⟩}
```

The argument is the internal command used to build the group header (which will then be appended to main internal command containing the glossary code). This hook may be redefined to insert any additional code before the heading. Use `\preto#1{`⟨*content*⟩`}` if you want to insert ⟨*content*⟩ before the header and `\appto#1{`⟨*content*⟩`}` if you want to insert ⟨*content*⟩ after the header. (You can reference the entry label with `\glscurrententrylabel` and the current hierarchical level with `\glscurrententrylevel` but make sure they are expanded if they occur in ⟨*content*⟩.) For example, `\printunsrttable` redefines this hook to finish off the current row before the group header is added.

Example 140 is a modification of the above document that shows the sub-group headings:

140



```
\renewcommand*{\glsxtraddgroup}[2]{#2}
\printunsrtglossary
\renewcommand*{\glsxtraddgroup}[2]{%
  \ifnum\glscurrententrylevel<2 #2\fi}
\printunsrtglossary[type=another]
```

Example 140: Displaying unsorted glossaries with custom groups and sub-group headings

Glossary

Group 1

waterfowl
ant

Group 2

adder

Group 4

duck

zebra
aardvark

Group 1

gazelle

Group 2

mallard

Another Glossary

mallard

Group 1

aardvark

Group 3

zebra

Group 4

ant

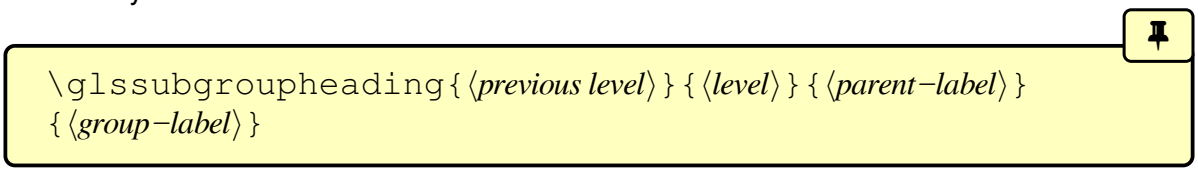
Group 2

duck

Note that the mallard entry (which has hierarchical level 2) has its group shown in the first glossary (where the group is formed for all levels) but not in the second glossary (where the redefinition of \glsxtraddgroup restricts group formation to just level 0 and level 1).

There's a small visual distinction between the group titles in different hierarchical levels in the above. The top-level (level 0) groups have the title centred, whereas the sub-groups have their titles indented by the same amount as the corresponding sub-entries. This is due to the glossary style. Other styles may use the same formatting for all hierarchical levels.

The glossary styles provided with glossaries−extra and the base styles patched by glossaries−extra−stylemods all redefine:

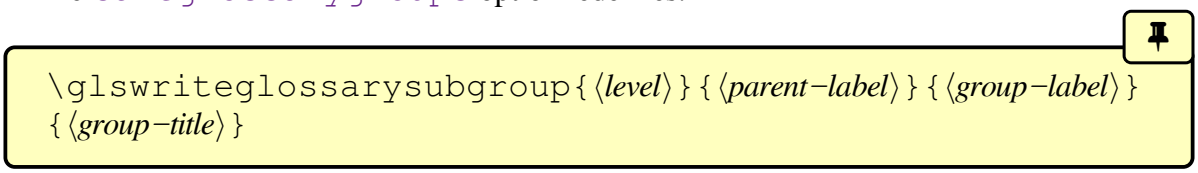


\glssubgroupheading{⟨*previous level*⟩}{⟨*level*⟩}{⟨*parent−label*⟩}{⟨*group−label*⟩}

to format the sub-group headings in a manner applicable to the style. For example, styles that don't show sub-entry names typically redefine this command to do nothing.

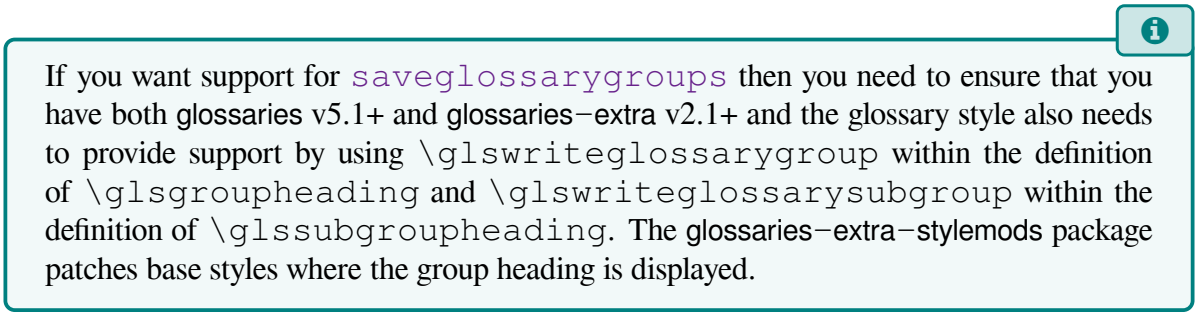
A default definition that simply does \glsgroupheading{⟨*group-label*⟩} is automatically initialised by \setglossarystyle (via \glsxtrpreglossarystyle) to allow for styles that don't redefine this command. The first two arguments refer to the hierarchical level, where ⟨*previous level*⟩ is the level of the previous group and ⟨*level*⟩ is the level of this new sub-group. The ⟨*parent-label*⟩ is the label of the current entry's parent, where the current entry is the first entry of the sub-group that immediately follows the heading.

The saveglossarygroups option redefines:



\glswriteglossarysubgroup{⟨*level*⟩}{⟨*parent−label*⟩}{⟨*group−label*⟩}{⟨*group−title*⟩}

This command is provided for use by \glssubgroupheading. The arguments ⟨*level*⟩, ⟨*parent-label*⟩ and ⟨*group-label*⟩ are the same as those for \glssubgroupheading. The ⟨*group-title*⟩ argument is the group's title.



If you want support for saveglossarygroups then you need to ensure that you have both glossaries v5.1+ and glossaries−extra v2.1+ and the glossary style also needs to provide support by using \glswriteglossarygroup within the definition of \glsgroupheading and \glswriteglossarysubgroup within the definition of \glssubgroupheading. The glossaries−extra−stylemods package patches base styles where the group heading is displayed.

The bookindex style defines \glssubgroupheading to use the style's associated command \glsxtrbookindexformatsubheader. This can be redefined as required. For example, the following uses the parent entry's hierarchical information:

```
\renewcommand*{\glsxtrbookindexformatsubheader}[5]{%
 \ifnum#2>1\relax
  \glstreesubsubitem
   \glstreegroupheaderfmt{\GlsXtrhiername{#3} / #5}%
 \else
  \glstreesubitem
   \glstreegroupheaderfmt{\GlsXtrhiername{#3} / #5}%
 \fi
}
```

The above examples are contrived and demonstrate the need to define entries in a sensible order to achieve a sensible glossary with `\printunsrtglossary`. If you want to use this approach to display a glossary, you would need to make sure that you take care with the order that you define entries. This can be quite tedious for a large number of entries.

Example 141 instead uses `bib2gls`. This means that the entry data needs to be provided in a `bib` file. For example, the file `animalfamilies.bib` might contain: 141

```
@index{waterfowl,user1={Anseriformes}}
@index{ant,user1={Formicidae}}
@index{adder,user1={Vipera berus}}
@index{duck,parent={waterfowl},user1={Anatidae}}
@index{zebra,user1={Hippotigris}}
@index{aardvark,user1={Orycteropus afer}}
@index{gazelle,user1={Gazella}}
@index{mallard,parent={duck},user1={Anas
platyrhynchos}}
```

I've included some additional information stored in the `user1` field that wasn't in the earlier examples. The document needs to use the `record` option and `\GlsXtrLoadResources` in order for it to work properly with `bib2gls`:

```
\usepackage
 [record,stylemods=bookindex,style=bookindex]
 {glossaries-extra}
\newglossary*{another}{Another Glossary}
\GlsXtrLoadResources[
 selection=all,% select all entries
 src={animalfamilies},% identify bib file(s)
 sort=en-GB,% sort method
 secondary={la:user1:another}
```

```
% sort by user1 (Latin) & copy to `another'
]
\glsdefpostname{index}{
 (\emph{\glsentryuseri{\glscurrententrylabel}})}
\begin{document}
\printunsrtglossary
\printunsrtglossary[type=another]
\end{document}
```

If this code is saved in the file `myDoc.tex` then the build process is now:

```
pdflatex myDoc
bib2gls --group myDoc
pdflatex myDoc
```

The `--group` (or `-g`) switch is important as it instructs `bib2gls` to set the `group` field for the primary sort and the `secondarygroup` for the secondary sort. The primary sort will sort entries according to `en-GB` (British English). This can simply be set to `en` without a region. The secondary sort will resort the entries, but this time according to `la` (Latin) using the `user1` key as the sort value. The entry labels will then be copied to the custom `another` glossary.

The `glstex` file created by `bib2gls` (which will then be input by `\GlsXtrLoadResources` on the subsequent LaTeX run) essentially contains the following code:

```
\glsxtrsetgrouptitle{6881280}{W}
\glsxtrsetgrouptitle{5832704}{G}
\glsxtrsetgrouptitle{5373952}{A}
\glsxtrsetgrouptitle{7077888}{Z}
\glsxtrsetgrouptitle{another5373952}{A}
\glsxtrsetgrouptitle{another5767168}{F}
\glsxtrsetgrouptitle{another6356992}{O}
\glsxtrsetgrouptitle{another5898240}{H}
\glsxtrsetgrouptitle{another6815744}{V}
\glsxtrsetgrouptitle{another5832704}{G}

\longnewglossaryentry*{aardvark}{name={aardvark},
 user1={Orycteropus afer},group={5373952}}{}
\longnewglossaryentry*{adder}{name={adder},
 user1={Vipera berus},group={5373952}}{}
\longnewglossaryentry*{ant}{name={ant},
 user1={Formicidae},group={5373952}}{}
```

```
\longnewglossaryentry*{gazelle}{name={gazelle},
 user1={Gazella},group={5832704}}{}
\longnewglossaryentry*{waterfowl}{name={waterfowl},
 user1={Anseriformes},group={6881280}}{}
\longnewglossaryentry*{duck}{name={duck},
 parent={waterfowl},user1={Anatidae},group={}}{}
\longnewglossaryentry*{mallard}{name={mallard},
 parent={duck},user1={Anas platyrhynchos},group={}}
{}
\longnewglossaryentry*{zebra}{name={zebra},
 user1={Hippotigris},group={7077888}}{}

\apptoglossarypreamble[another]
 {\renewcommand{\glsxtrgroupfield}{secondarygroup}}
\glsxtrcopytoglossary{waterfowl}{another}
\GlsXtrSetField{waterfowl}{secondarygroup}
{another5373952}
\glsxtrcopytoglossary{duck}{another}
\glsxtrcopytoglossary{mallard}{another}
\glsxtrcopytoglossary{ant}{another}
\GlsXtrSetField{ant}{secondarygroup}{another5767168}
\glsxtrcopytoglossary{gazelle}{another}
\GlsXtrSetField{gazelle}{secondarygroup}
{another5832704}
\glsxtrcopytoglossary{zebra}{another}
\GlsXtrSetField{zebra}{secondarygroup}
{another5898240}
\glsxtrcopytoglossary{aardvark}{another}
\GlsXtrSetField{aardvark}{secondarygroup}
{another6356992}
\glsxtrcopytoglossary{adder}{another}
\GlsXtrSetField{adder}{secondarygroup}
{another6815744}
```

It's more complicated than this as helper commands are provided to make it easier to customize and the entries will all have `category={index}` since they were defined with `@index`, but this is basically like the preamble in the earlier examples, except that the ordering and groups are more logical.

Note that in Example 141 the `group` and `secondarygroup` fields haven't been set for the child entries (duck and mallard). This is the default behaviour and it means that regardless of the definition you provide for `\glsxtraddgroup`, sub-groups won't be displayed. If you want those fields set for child entries, you need to use the `group-level` resource option. For example:

Example 141: Displaying sorted glossaries with groups using `bib2gls`

Glossary

A

aardvark (*Orycteropus afer*)
adder (*Vipera berus*)
ant (*Formicidae*)

G

gazelle (*Gazella*)

W

waterfowl (*Anseriformes*)
duck (*Anatidae*)
mallard (*Anas platyrhynchos*)

Z

zebra (*Hippotigris*)

Another Glossary

A

waterfowl (*Anseriformes*)
duck (*Anatidae*)
mallard (*Anas platyrhynchos*)

F

ant (*Formicidae*)

G

gazelle (*Gazella*)

H

zebra (*Hippotigris*)

O

aardvark (*Orycteropus afer*)

V

adder (*Vipera berus*)

```
\GlsXtrLoadResources[
 selection=all,% select all entries
 group-level={<=1},% level 0 and 1
 src={animalfamilies},% identify bib file(s)
 sort=en-GB,% sort method
 secondary={la:user1:another}% sort again and copy
]
```

This will add support for level 0 (no parent) and level 1 (parent but no grandparent) entries. Deeper levels won't have support. The `--group` switch is still required.

8.4.2. Location Lists

The "unsrt" family of commands check for the existence of the `location` and `loclist` keys. These are both defined by the `record` option. (The `loclist` field is also used by `\makenoidxglossaries` but isn't defined as a key.)

The `location` field (if set) should contain the formatted location list. This is checked first and used if not empty. Otherwise the `loclist` field (if set) is used, but that will use the same method as `\printnoidxglossary` to format, which doesn't compact consecutive locations.

It's possible to choose a different field for the formatted location list by redefining:

`\GlsXtrLocationField` *initial:* `location`

This should expand to the internal field label. If the field is not the default `location` then the test for `loclist` is omitted.

Whichever field is used, the formatted location list is passed to the appropriate glossary style command (`\glossentry` or `\subglossentry`) encapsulated with `\glossaryentrynumbers`.

If there's no location field or if the tested fields are empty, then an empty argument (with no `\glossaryentrynumbers` encapsulator) is passed to the glossary style command. In this case, the `nonumberlist` option is redundant as there's no location list to suppress.

8.4.3. Advanced Commands

To provide a better understanding of how filtered and inner glossaries work, it's useful to understand the difference between `\printglossary` (used with `makeindex` and `xindy`) and `\printunsrtglossary` (used with `bib2gls`).

In the first case, `makeindex` or `xindy` is used to create a file that contains content in the form:

```
\glossarysection[\glossarytoctitle]{\glossarytitle}%
\glossarypreamble
\begin{theglossary}\glossaryheader
⟨content⟩
\end{theglossary}\glossarypostamble
```

where ⟨*content*⟩ contains lines such as:

```
\glsgroupheading{⟨group label⟩}\relax \glsreseteentrylist
\glossentry{⟨entry label⟩}{⟨location list⟩}%
\subglossentry{⟨level⟩}{⟨entry label⟩}{⟨location list⟩}%
```

The group headings (see §8.4.1) are typeset using `\glsgroupheading`. Top-level entries are typeset with `\glossentry` and child entries are typeset with `\subglossentry` where ⟨*level*⟩ indicates the hierarchical level. Both `makeindex` and `xindy` order the items so that the child entries are placed immediately after the corresponding parent entry.

The `\printglossary` command essentially does:

```
⟨Set default title and style.⟩
\bgroup
   ⟨Initial setup.⟩
   ⟨Input the file created by makeindex or xindy.⟩
\egroup
```

The initial setup part sets the glossary style (which determines the definitions of theglossary, `\glossaryheader`, `\glsgroupheading`, `\glossentry` and `\subglossentry`), assigns the title (`\glossarytitle` and `\glossarytoctitle`) and defines `\currentglossary`. (There is some other stuff done both before and after the file is input, but that's not relevant here.)

In the case of `bib2gls`, there isn't a glossary file to input. Instead, `bib2gls` is used to create a file that contains the entry definitions, which is input in the document preamble (via `\GlsXtrLoadResources`). The entries are defined in the required order and use internal fields to store the indexing information (such as the group label and location lists). Now `\printunsrtglossary` is used to display the glossary, which essentially does:

```
⟨Set default title and style.⟩
\bgroup
   ⟨Initial setup.⟩
   \glossarysection
```

```
    [\glossarytoctitle]{\glossarytitle}%
  \glossarypreamble
  ⟨Construct internal control sequence containing glossary content.⟩
  ⟨Expand internal control sequence.⟩
  \glossarypostamble
\egroup
```

The ⟨*initial setup*⟩ is the same as for `\printglossary`. The key difference here is that there's no file containing the typeset glossary that can be simply input. Instead it's necessary to iterate over the glossary's internal label list. Some of the glossary styles use a tabular-like environment (such as longtable, which is used by the long styles). It's always problematic having a loop inside a tabular context so `\printunsrtglossary` by-passes the problem by moving the loop outside of the theglossary environment. The command iterates over all entry labels (in the order in which they were added to the glossary) and constructs an internal control sequence (`\@glsxtr@doglossary`), which ends up containing:

```
\begin{theglossary}\glossaryheader
⟨begin code⟩
⟨content⟩
⟨end code⟩
\end{theglossary}
```

Note that `\glsresetentrylist` has been removed in v1.50 since it's generally unnecessary with `bib2gls` and causes interference with tabular styles.

The ⟨*begin code*⟩ can be inserted just after `\begin{theglossary}` by the command:

```
\printunsrtglossarypostbegin{⟨internal cs⟩}
```

This does nothing by default (so ⟨*begin code*⟩ will be omitted). If you still need to have `\glsresetentrylist` at the start, you can redefine this hook as follows:

```
\renewcommand*{\printunsrtglossarypostbegin}[1]{%
 \appto#1{\glsresetentrylist}%
}
```

The ⟨*end code*⟩ can be inserted just before `\end{theglossary}` by the command:

```
\printunsrtglossarypreend{⟨internal cs⟩}
```

This does nothing by default (so ⟨*end code*⟩ will be omitted). (These two hooks are only used in `\printunsrtglossary` not in by `\printunsrtinnerglossary` or printunsrtglossarywrap.) For example, `\printunsrttable` redefines the end hook to finish off the final row.

In both hooks, the argument will be `\@glsxtr@doglossary` and, in both cases, you need to use `\appto` within the definition in order to insert ⟨*begin code*⟩ and ⟨*end code*⟩ in the correct place. If you use `\preto`, the code will end up at the start, before `\begin{theglossary}`

The ⟨*content*⟩ in this case is different as it doesn't explicitly contain `\glossentry` and `\subglossentry` but instead uses an internal handler that just takes the entry label as the argument. The `\glsgroupheading{⟨group label⟩}` command is inserted whenever a top-level entry has the group field set to a label that's different to the previous top-level entry's group field (and, if supported, sub-groups are similarly inserted with `\glssubgroupheading`, see §8.4.1). So the content is in the form:

```
\glsgroupheading{⟨group label⟩}%
\⟨internal cs handler⟩{⟨entry label⟩}%
\⟨internal cs handler⟩{⟨entry label⟩}%
...
\glsgroupheading{⟨group label⟩}%
\⟨internal cs handler⟩{⟨entry label⟩}%
\⟨internal cs handler⟩{⟨entry label⟩}%
...
```

There are hooks and commands available for use within those hooks that may be adjusted to customize the way the glossary is displayed. These are described below.

At each iteration (while the glossary content is being constructed), the following steps are performed:

1. Store the current entry label in `\glscurrententrylabel`.
2. If `\glscurrententrylabel` is empty, skip this iteration.
3. Define placeholder commands:

   ```
   \glscurrententrylevel
   ```

   This is set to the current entry's hierarchical level (taking `leveloffset` and `flatten` options into account);

```
\glscurrenttoplevelentry
```

This is set to the current entry label if `\glscurrententrylevel` is 0 (that is, it expands to the most recent top-level entry, allowing for `flatten` and `leveloffset`);

```
\glscurrentrootentry
```

This is set to the current entry label if `flatten=true` or if the current entry doesn't have a parent (that is, it expands to the most recent top-level entry, allowing for `flatten` but not `leveloffset`).

4. Perform the entry process hook:

```
\printunsrtglossaryentryprocesshook{⟨entry-label⟩}
```

This does nothing by default. Within the definition of this hook, you may use:

```
\printunsrtglossaryskipentry
```

This will cause the remainder of the current iteration to be skipped, which will prevent the current entry from being shown in the glossary.

5. If `groups=true`, use `\glsxtraddgroup` (see §8.4.1) to append the top-level group heading (`\glsgroupheading`) or the sub-group heading (`\glssubgroupheading`) to `\@glsxtr@doglossary`.

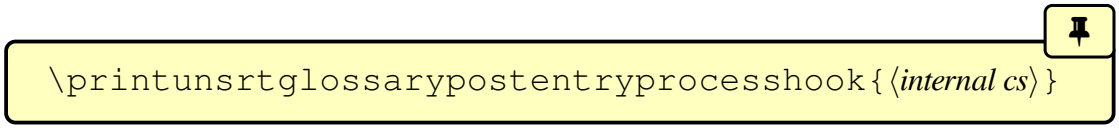
6. Perform the pre-entry process hook:

```
\printunsrtglossarypreentryprocesshook{⟨internal cs⟩}
```

The argument will be `\@glsxtr@doglossary`. This may be used to insert any additional content before the entry (use `\appto#1{⟨content⟩}`). (The entry label can be referenced with `\glscurrententrylabel` but make sure it's expanded if it occurs in ⟨*content*⟩.) For example, `\printunsrttable` redefines this hook to insert and `\tabularnewline` between blocks.

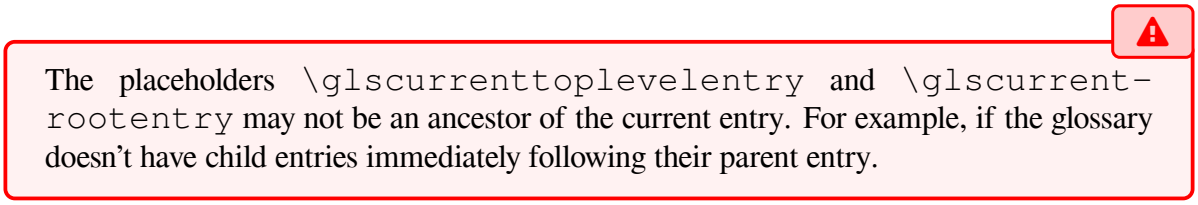
7. Append `\⟨internal cs handler⟩{⟨entry label⟩}` to `\@glsxtr@doglossary`.

8. Perform the post-entry process hook:



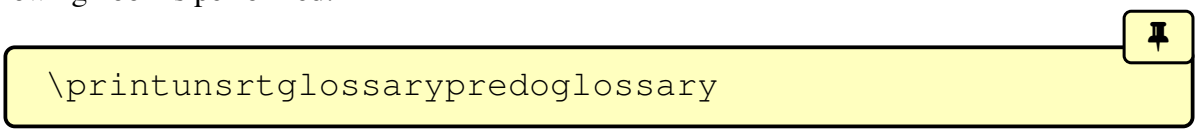
```
\printunsrtglossarypostentryprocesshook{⟨internal cs⟩}
```

The argument will be `\@glsxtr@doglossary`. This may be used to append any additional content after the entry (use `\appto#1{⟨content⟩}`). (The entry label can be referenced with `\glscurrententrylabel` but make sure it's expanded if it occurs in ⟨*content*⟩.) For example, `\printunsrttable` redefines this hook to reset the block index if the end of a row has been reached.



The placeholders `\glscurrenttoplevelentry` and `\glscurrentrootentry` may not be an ancestor of the current entry. For example, if the glossary doesn't have child entries immediately following their parent entry.

Once the glossary construction (`\@glsxtr@doglossary`) has been completed, the following hook is performed:



```
\printunsrtglossarypredoglossary
```

This does nothing by default. You can redefine this to show the definition of `\@glsxtr@doglossary` for debugging purposes:

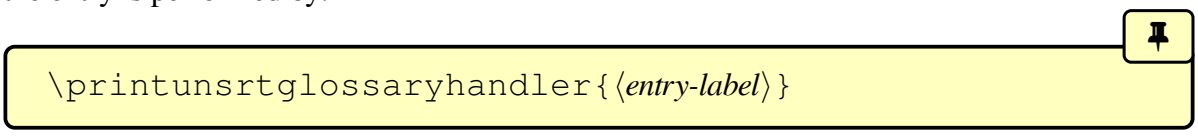
```
\renewcommand{\printunsrtglossarypredoglossary}{%
 \csshow{@glsxtr@doglossary}}
```

This will interrupt the LaTeX run and display the definition in the transcript.

The handler command `\`⟨*internal cs handler*⟩ performs the following:

```
\protected@xdef\glscurrententrylabel{⟨entry-label⟩}%
\printunsrtglossaryhandler\glscurrententrylabel
```

This stores the entry's label in `\glscurrententrylabel` (which allows it to be referenced in style hooks, such as the post-name hook or post-description hook). Note that it uses a global definition to avoid scoping issues caused with tabular-like styles. The main handling of the entry is performed by:



```
\printunsrtglossaryhandler{⟨entry-label⟩}
```

This is simply defined to use:

\glsxtrunsrtdo{⟨*entry-label*⟩}

This displays the entry according to the current glossary style, taking the hierarchical level into account (as given by \glscurrententrylevel).

The following are additional commands that may be useful in the above hooks.

\glsxtriflabelinlist{⟨*label*⟩}{⟨*label-list*⟩}{⟨*true*⟩}{⟨*false*⟩}

Does ⟨*true*⟩ if the given label is in the given comma-separated list of labels, otherwise does ⟨*false*⟩. The label and list are fully expanded.

\ifglsxtrprintglossflatten ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iffalse

This conditional is set by the flatten option and can be used to test if the option has been set.

Example 142 skips all entries that have the category set to symbol:

142

```
\usepackage[style=index]{glossaries-extra}
\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{pi}{
 name={\ensuremath{\pi}},description={},
 category={symbol}}
\newglossaryentry{aardvark}{
 name={aardvark},description={}}
\newglossaryentry{alpha}{
 name={\ensuremath{\alpha}},description={},
 category={symbol}}
\begin{document}
\renewcommand{\printunsrtglossaryentryprocesshook}[1]{%
 \glsifcategory{#1}{symbol}%
 {\printunsrtglossaryskipentry}%
 {}%
}
\printunsrtglossary
\end{document}
```

Example 142: Filtering by category

Glossary

ant
aardvark

8.4.3.1. Inner Glossaries

See also Gallery: Inner or Nested Glossaries.[a]

[a] dickimaw-books.com/gallery/index.php?label=bib2gls-inner

It's possible you may want to combine multiple glossaries sequentially, as sub-blocks of a single list. The inner part of `\printunsrtglossary` can be created with:

`\printunsrtinnerglossary[`⟨*options*⟩`]{`⟨*pre-code*⟩`}{`⟨*post-code*⟩`}`

This can't be used on its own, as it only forms a fragment. It doesn't include the section header, style initialisation, preamble, theglossary environment, header and postamble.

As with `\printunsrtglossary`, `\printunsrtinnerglossary` constructs an internal control sequence containing the content, but it adds scoping to localise the effects of any supplied options. So it essentially does:

```
\begingroup
   ⟨Initial setup (process options).⟩
   ⟨pre-code⟩
   ⟨Construct internal control sequence containing glossary content.⟩
   ⟨Expand internal control sequence.⟩
   ⟨post-code⟩
\endgroup
```

There are two ways this command may be used.

`\begin{`**`printunsrtglossarywrap`**`}[`⟨*options*⟩`]`
⟨*content*⟩
`\end{`**`printunsrtglossarywrap`**`}`

The printunsrtglossarywrap environment takes one optional argument that uses the same keys as `\printunsrtglossary` (see §8.3). Note that in this case the `type` key simply provides a title (if one has been assigned to that glossary). It doesn't indicate the content. There's no point using both `type` and `title`.

The start of the environment sets up the glossary style and does the header:

```
⟨Set default title and style.⟩
⟨Initial setup.⟩
\glossarysection
 [\glossarytoctitle]{\glossarytitle}%
\glossarypreamble
\begin{theglossary}\glossaryheader\glsreseteentrylist
```

The end of this wrapper environment ends theglossary and does the postamble:

```
\end{theglossary}\glossarypostamble
```

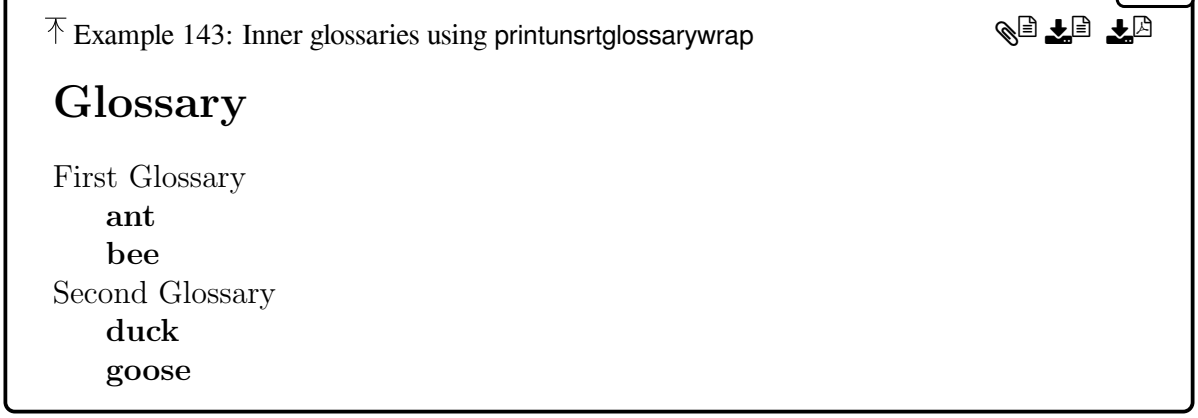
Note that the `\printunsrtglossarypostbegin`, `\printunsrtglossarypreend` and `\printunsrtglossarypredoglossary` hooks aren't used.

Example 143 has two glossaries but displays them as inner glossaries:

143

```
\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{bee}{name={bee},description={}}
\newignoredglossary*{other}
\newglossaryentry{duck}{name={duck},description={}}
\newglossaryentry{goose}{name={goose},description={}}
\begin{document}
\begin{printunsrtglossarywrap}[style=index]
\glstreeitem First Glossary
\printunsrtinnerglossary[leveloffset=1]{}{}
\glstreeitem Second Glossary
\printunsrtinnerglossary[type=other,leveloffset=1]{}{}
\end{printunsrtglossarywrap}
\end{document}
```

Example 143: Inner glossaries using printunsrtglossarywrap



The other way that \printunsrtinnerglossary can be used is within \printunsrtglossary. The handler function described in §8.4.3 that's used to process each entry to be displayed in the glossary, is defined as:

```
\newcommand{\printunsrtglossaryhandler}[1]{%
 \glsxtrunsrtdo{#1}}
```

It's possible to redefine this handler command so that it also displays an inner glossary.

Example 144 has the terms "pictograph" and "Greek symbol" in the main glossary. Two ignored glossaries are created (which don't require a title) where the glossary label matches an entry label in the main glossary. 144

```
\usepackage{fontawesome}
\usepackage[style=tree]{glossaries-extra}
\newglossaryentry{pictograph}{name={pictograph},
 description={picture or symbol representing a word
or phrase}}
\newglossaryentry{mathgreek}{name={Greek symbol},
 description={mathematical constants or functions}}
\newignoredglossary*{pictograph}
\newignoredglossary*{mathgreek}
\newglossaryentry{cut}{type={pictograph},
 name={\faCut},
 description={cut}}
\newglossaryentrypaste{type={pictograph},
name={\faPaste},
 description={paste}}
\newglossaryentry{alpha}{type={mathgreek},
 name={\ensuremath{\alpha}},
 description={alpha}}
```

```
\newglossaryentry{beta}{type={mathgreek},
 name={\ensuremath{\beta}},
 description={beta}}
\begin{document}
\newcommand{\nestedhandler}[1]{%
 \glsxtrunsrtdo{#1}%
 \ifglossaryexists*{#1}%
 {%
   \printunsrtinnerglossary[type={#1},
    leveloffset={++1},groups=false]{}{}%
 }%
 %
}
\printunsrtglossary*{%
 \let\printunsrtglossaryhandler\nestedhandler}
\end{document}
```

This creates a custom command `\nestedhandler` that can be used as the handler to create nested glossaries. After each item in the glossary, if the entry's label matches the label of a defined glossary, that glossary is displayed with its hierarchical level incremented by 1, which creates the illusion of child entries.

Example 144: Nested glossaries

Glossary

pictograph picture or symbol representing a word or phrase
- ✂ cut
- 📋 paste

Greek symbol mathematical constants or functions
- α alpha
- β beta

8.4.3.2. Per-Unit Glossaries

If you are using `bib2gls` then it's possible to only list entries that match a particular counter value. For example, you may want a mini-glossary at the start of a section that only lists the entries that have been recorded in that section. This can be done by using the handler to skip entries that don't have a matching record. It can also be implemented with record counting, as shown in Example 161 in §11.5.

It's also possible to make each indexing instance automatically make a note of a particular counter using:

```
\GlsXtrRecordCounter{⟨counter-name⟩}
```

(This doesn't correspond to a `bib2gls` record. That's dealt with by the indexing that comes first.)

This command may only be used in the preamble (with `record`) and indicates that whenever an entry is indexed, the following line should be added to the `aux` file:

```
\glsxtr@counterrecord{⟨entry-label⟩}{⟨counter⟩}{⟨value⟩}
```

where ⟨*value*⟩ is given by `\the`⟨*counter*⟩. On the next LaTeX run, this information is picked up from the `aux` file and the information is added to the `record.`⟨*counter*⟩ field (stored as an etoolbox internal list). This internal command is only used in the `aux` file and has a user-level hook:

```
\glsxtrAddCounterRecordHook{⟨entry-label⟩}{⟨counter⟩}{⟨value⟩}
```

This does nothing by default. If you want to redefined this, the redefinition must be placed in the document preamble before the `aux` file is input.

There are two ways of skipping an entry. The first is to redefine `\printunsrtglossaryentryprocesshook` to perform the test and use `\printunsrtglossaryskipentry` to skip an unwanted entry (as illustrated earlier). The second is to perform the test in `\printunsrtglossaryhandler`. The first method is the better option for large lists that may contain group headers. The example below uses the second method.

The file `myentries.bib` contains the following:

```
@symbol{pi,name={\ensuremath{\pi}},
description={ratio of the length of the
circumference of a circle to its diameter}}
@symbol{root2,name={\ensuremath{\surd2}},
description={Pythagoras' constant}}
@symbol{zeta3,name={\ensuremath{\zeta(3)}},
description={Ap\'ery's constant}}
@symbol{zero,name={0},
description={nothing or nil}}
@symbol{one,name={1},
description={single entity, unity}}
```

The document redefines the handler to only show entries in the current section:

```
\usepackage[record,stylemods,style=index]
{glossaries-extra}
\GlsXtrRecordCounter{section}
\GlsXtrLoadResources[src={myentries}]
\begin{document}
\renewcommand{\printunsrtglossaryhandler}[1]{%
  \glsxtrfieldxifinlist{#1}{record.section}
   {\thesection}
  {\glsxtrunsrtdo{#1}}%
  {}%
}
\section{Sample}
\printunsrtglossary
This section discusses \gls{pi}, \gls{root2} and
\gls{zeta3}.

\section{Another Sample}
\printunsrtglossary
This section discusses \gls{one}, \gls{pi} and
\gls{zero}.
\end{document}
```

If the document is saved in the file `myDoc.tex` then the build process is:

```
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
```

The first LaTeX run adds the records to the `aux` file for `bib2gls` to pick up, but also adds the `\glsxtr@counterrecord` lines (which `bib2gls` ignores) that setup the `record.section` list field for the given entry.

This means that `\glsxtrfieldxifinlist` can be used to determine whether or not the current section number (`\thesection`) is in the list. If it is, then the entry is displayed in the current glossary style using the default `\glsxtrunsrtdo`. Otherwise nothing is displayed.

The following command is provided that performs something similar:

```
\printunsrtglossaryunit[⟨options⟩]{⟨counter-name⟩}
```

This is equivalent to:

```
\printunsrtglossary*[type=\glsdefaulttype,#1]{%
  \printunsrtglossaryunitsetup{#2}%
}
```

This initialises the hook via:

```
\printunsrtglossaryunitsetup{⟨counter-name⟩}
```

This is essentially does:

```
% redefine handler to only show entries with a match:
\renewcommand{\printunsrtglossaryhandler}[1]{%
  \glsxtrfieldxifinlist{#1}{record.⟨counter-name⟩}
    {\the⟨counter-name⟩}
  {\glsxtrunsrtdo{#1}}%
  {}%
}%
⟨assign target name prefixes (see below)⟩
% suppress section header:
\renewcommand*{\glossarysection}[2][]{}%
% append vertical space after the glossary:
\appto\glossarypostamble{%
 \printunsrtglossaryunitpostskip}%
```

This is more complicated than the original example as it also suppresses the glossary section header and modifies the target name prefix. Additionally, the following is appended to the end of the glossary:

```
\printunsrtglossaryunitpostskip
```

This simply does:

```
\glspar\medskip\glspar
```

which creates a small vertical space. The target name prefix (`targetnameprefix`) is assigned as follows. If `\the⟨counter-name⟩` has been defined, the prefix is:

```
record.⟨counter-name⟩.\the⟨counter-name⟩.\@gobble
```

otherwise the prefix is:

```
record.⟨counter-name⟩.\the⟨counter-name⟩.\@gobble
```

The use of \@gobble at the end discards \glolinkprefix.

Example 145 is a modification of the above example that uses \printunsrtglossaryunit. I've added symbol-sort-fallback to sort by the description and a full glossary at the end of the document. 145

```
\usepackage[record,stylemods,style=index]
{glossaries-extra}
\GlsXtrRecordCounter{section}
\GlsXtrLoadResources[src={myentries},
 symbol-sort-fallback=description
]
\begin{document}
\section{Sample}
\printunsrtglossaryunit{section}
This section discusses \gls{pi}, \gls{root2} and
\gls{zeta3}.

\section{Another Sample}
\printunsrtglossaryunit{section}
This section discusses \gls{one}, \gls{pi} and
\gls{zero}.

\printunsrtglossaries
\end{document}
```

The build process is the same as before:

```
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
```

Note that all glossaries show the location lists, which all contain the page number 1, since the example document is only one page long.

Other variations include creating a secondary glossary that's ordered differently for the mini-glossaries. For example:

```
\newignoredglossary*{glossary2}
\GlsXtrLoadResources[src={\jobname},
```

⇑ Example 145: Sub-glossary for a given counter value

1 Sample

$\zeta(3)$ Apéry's constant 1
$\sqrt{2}$ Pythagoras' constant 1
π ratio of the length of the circumference of a circle to its diameter 1

This section discusses π, $\sqrt{2}$ and $\zeta(3)$.

2 Another Sample

0 nothing or nil 1
π ratio of the length of the circumference of a circle to its diameter 1
1 single entity, unity 1

This section discusses 1, π and 0.

Glossary

$\zeta(3)$ Apéry's constant 1
0 nothing or nil 1
$\sqrt{2}$ Pythagoras' constant 1
π ratio of the length of the circumference of a circle to its diameter 1
1 single entity, unity 1

```
 symbol-sort-fallback=description,
 secondary=use:glossary2
]
```

This orders the secondary glossary according to use (the first record for the entire document not for the given unit). The mini-glossaries will then need the `type` option:

```
\printunsrtglossaryunit[type=glossary2]{section}
```

There is an alternative method that ensures the mini-glossaries are ordered by use within the section. This can be done by redefining `\glsxtrAddCounterRecordHook` to create a glossary for each unit (instead of using a secondary glossary):

```
\renewcommand{\glsxtrAddCounterRecordHook}[3]{%
 \provideignoredglossary{#2.#3}%
 \glsxtrcopytoglossary*{#1}{#2.#3}%
}
```

(Remember this needs to be done in the preamble, before the `aux` file is input.)

This creates a glossary with the label ⟨*counter*⟩`.`⟨*value*⟩, if it's not already defined, and adds the entry's label to it. This means that this glossary will only contain the entries for the matching ⟨*counter*⟩ and ⟨*value*⟩, and the entry labels are in the order they were added to the `aux` file.

The glossary needs to be set appropriately. For example:

```
\printunsrtglossaryunit[type=section.\thesection]
{section}
```

There's now no filtering required, but `\printunsrtglossaryunit` is still useful as it automatically suppresses the section header, alters the hyperlink prefix and adds extra spacing after the glossary. However, if you prefer, you can simply do something like:

```
\printunsrtglossary*[type=section.\thesection,
 target=false]
\renewcommand*{\glossarysection}[2][]{}
\printunsrtglossaryunitpostskip
```

This is done in Example 146.

146

⤒ Example 146: Sub-glossary for a given counter value ordered by use in the section

1 Sample

π ratio of the length of the circumference of a circle to its diameter 1
$\sqrt{2}$ Pythagoras' constant 1
$\zeta(3)$ Apéry's constant 1

This section discusses π, $\sqrt{2}$ and $\zeta(3)$.

2 Another Sample

1 single entity, unity 1
π ratio of the length of the circumference of a circle to its diameter 1
0 nothing or nil 1

This section discusses 1, π and 0.

Glossary

$\zeta(3)$ Apéry's constant 1
0 nothing or nil 1
$\sqrt{2}$ Pythagoras' constant 1
π ratio of the length of the circumference of a circle to its diameter 1
1 single entity, unity 1

8.5. Standalone Entry Items

It may be that you don't want a list but would rather display entry details throughout the document. You can simply do `\glsentryname` followed by `\glsentrydesc`. (Remember that if you don't want a sorted list, use `sort=none` or `sort=clear` to skip the preprocessing of the `sort` field.) For example, in the preamble provide a custom command to display the entry's name and description:

```
\newcommand{\displayterm}[1]{%
 \par\medskip\par\noindent
 Definition: \glsentryname{#1}.\par
 \glsentrydesc{#1}
 \par\medskip
}
```

define your entries, for example:

```
\newglossaryentry{function}{name={function},
  description={a relation or expression
involving variables}
}
```

and then later in the text:

```
\displayterm{function}
```

However, if may be that you want to use hyperref and have commands like `\gls` link back to the place where the term is described. Instead of using `\glsentryname` use:

```
\glsxtrglossentry{⟨entry-label⟩}
```

where ⟨*entry-label*⟩ is the entry's label. If sentence case is required, use:

```
\Glsxtrglossentry{⟨entry-label⟩}
```

Since \glossentryname checks the glossname attribute, it is possible to use \glsxtrglossentry with the glossname attribute set to firstuc to apply sentence case. However, \Glsxtrglossentry integrates better in section headings.

These are designed to behave much like the way the name is displayed in the glossary. They perform the following:

- Defines \glscurrententrylabel to the entry's label. This is usually done at the start of the glossary style commands \glossentry and \subglossentry and may be used by hooks, such as the post-name hook. Here the definition is localised so that it's only available for use in \glossentryname.
- Defines \currentglossary to the entry's glossary type. This is usually done at the start of commands like \printglossary and may be used by style hooks. Here the definition is localised so that it's only available for use in \glsentryitem and \glssubentryitem. The value is obtained by fully expanding:

```
\GlsXtrStandaloneGlossaryType
```

  which defaults to the value of the type field for the current entry.
- Increments and display the entry counters if the entrycounter or subentrycounter package options are set. If the entry doesn't have a parent, then this does:

```
\glsentryitem{⟨entry-label⟩}
```

  otherwise it does:

```
\GlsXtrStandaloneSubEntryItem{⟨entry-label⟩}
```

  which defaults to \glssubentryitem{⟨entry-label⟩} if the entry has a parent but not a grandparent.

  This reflects the behaviour of the predefined hierarchical styles. A bug in pre-version 1.31 used \glsentryitem for all child levels, which doesn't match the hierarchical glossary styles. If you want to restore this behaviour, just do:

```
\renewcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
 \glssubentryitem{#1}}
```

- Sets the hyper-target if supported (using \glstarget) and displays the entry name using:

```
\GlsXtrStandaloneEntryName{⟨entry-label⟩}
```

  which uses \glstarget{⟨entry-label⟩}{\glossentryname{⟨entry-label⟩}} by default. Or, for the sentence case version:

```
\GlsXtrStandaloneEntryNameFirstUc{⟨entry-label⟩}
```

  which uses \Glossentryname instead.

  Remember that \glossentryname uses \glsnamefont or picks up the style from category attributes such as glossnamefont. This can result in duplicate targets if you use both standalone commands and display the glossary. In which case, you can redefine \glstarget to use \glsxtrtarget, which will ensure that the first target will be the one that takes precedence.

If you have used \nopostdesc or \glsxtrnopostpunc in any of your description fields, you can use:

```
\glsxtractivatenopost
```

to make these commands behave as they normally do within a glossary. This needs to be placed before:

```
\glossentrydesc{⟨entry-label⟩}\glspostdescription
```

and scoped. Note that \glsnonextpages and \glsnextpages have no effect outside of the glossary and are not intended for use in a standalone context.

It's also possible to select a different field (rather than using name):

```
\glsxtrglossentryother{⟨header⟩}{⟨entry-label⟩}{⟨field-label⟩}
```

Again, there is a sentence case variant:

```
\Glsxtrglossentryother{⟨header⟩}{⟨entry-label⟩}{⟨field-label⟩}
```

If the ⟨*header*⟩ argument is not empty, you will need to ensure that it has the appropriate casing applied as it won't be changed automatically.

The ⟨*field-label*⟩ must be given using its internal field label. The ⟨*header*⟩ argument is the code to pass to the third argument of `\glsxtrtitleorpdforheading`. It may be left empty in which case the default is determined as follows:

- If the command `\glsxtrhead`⟨*field-label*⟩ is defined (for example, `\glsxtrhead-short` if ⟨*field-label*⟩ is `short`, see §5.3.3), then ⟨*header*⟩ is `\glsxtrhead`⟨*field-label*⟩`{`⟨*entry-label*⟩`}`.
- Otherwise ⟨*header*⟩ is simply the field value.

The `\glsxtrglossentryother` command internally uses

```
\GlsXtrStandaloneEntryOther{⟨entry-label⟩}{⟨field-label⟩}
```

which will use `\glossentrynameother{`⟨*entry-label*⟩`}{`⟨*field-label*⟩`}`. The sentence case variant uses:

```
\GlsXtrStandaloneEntryOtherFirstUc{⟨entry-label⟩}{⟨field-label⟩}
```

which uses `\Glossentrynameother` instead.

If you have loaded the glossaries−accsupp package (through the `accsupp` option) then accessibility support will be provided if there's a corresponding command:

```
\gls⟨field-label⟩accessdisplay{⟨text⟩}{⟨entry-label⟩}
```

(for example, `\glssymbolaccessdisplay`).

This means that my custom command can be changed to:

```
\newcommand{\displayterm}[1]{%
 \par\medskip\par\noindent
 Definition: \glsxtrglossentry{#1}.\par
 \glsentrydesc{#1}
 \par\medskip
}
```

If I want numbered definitions, then I can use the package options `entrycounter` or `subentrycounter` and remove the colon:

```
\newcommand{\displayterm}[1]{%
 \par\medskip\par\noindent
 Definition \glsxtrglossentry{#1}.\par
```

```
  \glsentrydesc{#1}
  \par\medskip
}
```

The counter label uses a dot after the number by default but this can be changed to a colon:

```
\renewcommand*{\glsentrycounterlabel}{%
 \theglossaryentry:\space}
```

It's now possible to not only use `\gls` to link back to the definition but also use `\glsrefentry` to reference the counter and `\glsxtrpageref` to reference the page number.

If I want the description to behave more like it does in a glossary in need to make the following modification:

```
\newcommand{\displayterm}[1]{%
 \par\medskip\par\noindent
 Definition \glsxtrglossentry{#1}.\par
 \begingroup
  \glsxtractivatenopost
  \glossentrydesc{#1}\glspostdescription
 \endgroup
 \par\medskip
}
```

(Note the grouping to localise `\glsxtractivatenopost`.)

You can also use `\glsxtrglossentry` within section headings. For example:

```
\section{\glsxtrglossentry{function}}
```

If `\glsxtrglossentry` occurs in a section title and hyperref has been loaded, then `\glsxtrglossentry` will expand in the PDF bookmark as:

```
\GlsXtrStandaloneEntryPdfName{⟨entry-label⟩}
```

This defaults to `\glsentryname{⟨entry-label⟩}` The page headers and table of contents will use

```
\GlsXtrStandaloneEntryHeadName{⟨entry-label⟩}
```

which defaults to \glsxtrheadname{⟨*entry-label*⟩}. The sentence case \Glsxtrglossentry has similar commands:

```
\GlsXtrStandaloneEntryPdfNameFirstUc{⟨entry-label⟩}
```

for the PDF bookmark and

```
\GlsXtrStandaloneEntryHeadNameFirstUc{⟨entry-label⟩}
```

For example, if you want sentence case with \glsxtrglossentry instead of using \Glsxtrglossentry, then to ensure that the name is displayed in sentence case in the title, PDF bookmarks and heading:

```
\glssetcategoryattribute{general}{glossname}
{firstuc}
\renewcommand{\GlsXtrStandaloneEntryPdfName}[1]
{\Glsentryname{#1}}
\renewcommand{\GlsXtrStandaloneEntryHeadName}[1]
{\Glsentryname{#1}}
```

Note that this requires glossaries v4.50+ to ensure that \Glsentryname expands. An alternative is to use \Glsxtrusefield.

If \glsxtrglossentryother occurs in a section title and hyperref has been loaded, then \glsxtrglossentryother will expand in the PDF bookmark as:

```
\GlsXtrStandaloneEntryPdfOther{⟨entry-label⟩}{⟨field-label⟩}
```

This defaults to the value of the given field. The page headers and table of contents will use the ⟨*header*⟩ argument, if not empty, otherwise it will use:

```
\GlsXtrStandaloneEntryHeadOther{⟨entry-label⟩}{⟨field-label⟩}
```

This does \glsxtrhead⟨*field-label*⟩, if it exists, or otherwise it just does the value of the given field (which can be obtained with \glsxtrusefield).

The sentence case \Glsxtrglossentryother has corresponding commands:

```
\GlsXtrStandaloneEntryPdfOtherFirstUc{⟨entry-label⟩}{⟨field-label⟩}
```

for the PDF bookmark and:

\GlsXtrStandaloneEntryHeadOtherFirstUc{⟨*entry-label*⟩}{⟨*field-label*⟩}

for the page header and table of contents.

If you're using a page style or table of contents that doesn't use \markright or \markboth or \@starttoc then you need to insert \glsxtrmarkhook and \@glsxtrinmark at the start of the header or table of contents either scoped or afterwards cancelled with \@glsxtrnotinmark and \glsxtrrestoremarkhook, see §5.3.3.

8.6. Glossary Style Modifications

The glossaries−extra package redefines \setglossarystyle, and it now includes a hook that's performed before the style is set:

\glsxtrpreglossarystyle

This allows for new style commands that aren't provided by the base glossaries package to be initialised in the event that a style that doesn't redefine them is used. The default definition is:

```
\newcommand{\glsxtrpreglossarystyle}{%
  \renewcommand*{\glssubgroupheading}[4]{%
    \glsgroupheading{##4}}%
}
```

If you prefer a different default, you can redefine this command as appropriate. Bear in mind that if you want support for the saveglossarygroups package option then the definition of \glssubgroupheading will need to include \glswriteglossarysubgroup.

The commands \glossentryname and \glossentrydesc are modified to take into account the glossname, glossnamefont, glossdesc and glossdescfont attributes (see §10). This means you can make simple font or case-changing modifications to the name and description without defining a new glossary style.

The command \glossentrysymbol is modified to take into account the glosssymbolfont attribute. Note that, unlike the above, there's no corresponding attribute to change the case as it's usually not appropriate to change the case of a symbol (and for some symbols, such as pictographs, there's no concept of case). If \texorpdfstring has been defined \glossentrysymbol will be defined to do:

\texorpdfstring{⟨*TEX code*⟩}{⟨*PDF*⟩}

The ⟨*TEX code*⟩ part is robust and deals with the actual typesetting of the symbol. The ⟨*PDF*⟩ part is simply:

```
\glsentrypdfsymbol{⟨entry-label⟩}
```

which is defined to just do `\glsentrysymbol{⟨entry-label⟩}`. The chances are that the code in the `symbol` key won't be valid in the PDF bookmarks, so you can redefine `\glsentrypdfsymbol` to use a more appropriate field. (If you do redefine this command, remember that it needs to fully expand.)

For example, if you are using glossaries-accsupp, you could use the `symbolaccess` field:

```
\renewcommand{\glsentrypdfsymbol}[1]{%
 \glsentrysymbolaccess{#1}}
```

Alternatively, if you are using `bib2gls` you can use the TEX parser library to interpret a copy of the `symbol` field and use that. For example, with the resource options:

```
replicate-fields={symbol=user1},
interpret-fields=user1
```

This copies the value of the `symbol` field to the `user1` field (`replicate-fields`) and then replaces the value of the `user1` field with its interpreted value (`interpret-fields`).

This means you can then do:

```
\renewcommand{\glsentrypdfsymbol}[1]{%
 \glsentryuseri{#1}}
```

(You may need XeLaTeX or LuaLaTeX with this method.) This allows `\glossentrysymbol` to be used in a section heading with standalone definitions. See the `bib2gls` manual for further details about the TEX interpreter.

If you want to adapt a style to use another field instead of `name`, you can use:

```
\glossentrynameother{⟨entry-label⟩}{⟨field-label⟩}
```

This behaves just like `\glossentryname` (that is, it obeys the `glossname` attribute, uses either the `glossnamefont` attribute or `\glsnamefont` to format the text, and uses the post-name hook) but the text is obtained from the field given ⟨*field-label*⟩ instead of `name`. The ⟨*field-label*⟩ argument must be the internal field label (for example `desc` rather than `description`).

If you prefer to bypass the glossname attribute and always apply sentence case, then you can instead use:

```
\Glossentrynameother{⟨entry-label⟩}{⟨field-label⟩}
```

This behaves as \glossentrynameother but omits the glossname attribute check.

Similarly for all uppercase:

```
\GLOSSentrynameother{⟨entry-label⟩}{⟨field-label⟩}
```

(which uses \glsuppercase) or for title case:

```
\GlossEntryNameOther{⟨entry-label⟩}{⟨field-label⟩}
```

This internally uses \glsentrytitlecase to perform the case-change.

8.6.1. Pre- and Post-Name Hooks

The glossaries-extra package adds hooks to \glossentryname and \Glossentryname (which is used in glossary styles to display the entry's name). Similarly for \GLOSSentryname and \GlossEntryName (which are new to glossaries v4.59).

```
\glsxtrprenamehook{⟨entry-label⟩}
```

The pre-name hook is performed before the entry name is displayed (but after the abbreviation style is set for the entry's category, if applicable). Does nothing by default.

Unlike the post-name hook described below, the pre-name hook isn't included in the font change nor is it influenced by any case-change. It may be used, for example, to add a marker before the name.

If the pre-name hook includes a declaration, bear in mind that most styles don't apply a scope to the name and the post-name hook may need to be adjusted to cancel the effect.

```
\glsxtrpostnamehook{⟨entry-label⟩}
```

This is the main post-name hook, which implements additional hooks to allow for customisation. By default, \glsxtrpostnamehook checks the indexname attribute. If the attribute exists for the category to which the entry belongs, then the name is automatically indexed using:

```
\glsxtrdoautoindexname{⟨entry-label⟩}{indexname}
```

See §12 for further details.

The post-name hook `\glsxtrpostnamehook` will also use:

```
\glsxtrpostname⟨category⟩
```

if it exists. You can use `\glscurrententrylabel` to obtain the entry label with the definition of this command. For example, suppose you are using a glossary style the doesn't display the symbol, you can insert the symbol after the name for a particular category, say, the "symbol" category:

```
\newcommand*{\glsxtrpostnamesymbol}{\space
 (\glsentrysymbol{\glscurrententrylabel})}
```

For convenience, you can use:

```
\glsdefpostname{⟨category⟩}{⟨definition⟩}
```

This is simply a shortcut for:

```
\csdef{\glsxtrpostname⟨category⟩}{⟨definition⟩}
```

Note that it doesn't check if the command has already been defined.

The post-name hook also does:

```
\glsextrapostnamehook{⟨entry-label⟩}
```

(before `\glsxtrpostname⟨category⟩`) to allow for additional non-category related code. This does nothing by default.

8.6.2. Post-Description Hooks

The glossaries package provides the hook `\glspostdescription`, which is placed after the description in some of the predefined styles. The glossaries-extra-stylemods package modifies the predefined styles to ensure that they all use this hook. This provides a convenient way to make slight adjustments, such as appending content after the description, without having to define a custom glossary style.

The glossaries-extra package redefines `\glspostdescription` so that it includes the following hook:

\glsxtrpostdescription

This new hook simply performs the category post-description hook:

```
\newcommand*{\glsxtrpostdescription}{%
  \csuse{glsxtrpostdesc\glscategory
    {\glscurrententrylabel}}%
}
```

The punctuation that is automatically inserted with postdot or postpunc is placed after \glsxtrpostdescription, not before.

If you want to modify the hook for all entries (without affecting the postpunc or postdot options), then redefine \glsxtrpostdescription. If you want to adjust this hook according to the entry's category, then you can simply redefine the category post-description hook.

\glsxtrpostdesc⟨*category*⟩

Some common category post-description hooks are provided:

\glsxtrpostdescgeneral *initial: empty*

The post-description hook for the general category.

\glsxtrpostdescterm *initial: empty*

The post-description hook for the term category.

\glsxtrpostdescacronym *initial: empty*

The post-description hook for the acronym category.

\glsxtrpostdescabbreviation *initial: empty*

The post-description hook for the abbreviation category.

The above all do nothing by default. You can redefine them with \renewcommand or use:

\glsdefpostdesc{⟨*category*⟩}{⟨*definition*⟩}

This will define (or redefine) \glsxtrpostdesc⟨*category*⟩. The package options symbols, numbers and index provide corresponding category post-description hooks.

You can reference the current entry within these hooks using \glscurrententry-label, which is defined within the glossary (any of the \print...glossary commands) and also within the standalone commands, such as \glsxtrglossentry.

\glsxtrnopostpunc

Suppresses the post-description punctuation that is automatically inserted by package options postdot or postpunc.

The glossaries package provides \nopostdesc, which may be used in the description to suppress the post-description hook for that entry. This suppresses both the post-description punctuation and the additional \glsxtrpostdescription hook. If you only want to suppress to punctuation, then use \glsxtrnopostpunc instead.

The post-description hooks are implemented by \glspostdescription within the glossary style. If this command isn't used in the style, then the additional hooks won't be available.

\glsxtrrestorepostpunc

If this command is placed in the definition of \glsxtrpostdescription or added to the category post-link hook, then it will counter-act any use of \glsxtrnopostpunc to restore the post-description punctuation.

These commands have no effect outside of the glossary (except with standalone entries that use \glsxtractivatenopost and \glspostdescription, see §8.5).

8.6.3. Number (Location) List

The location list is now placed inside the argument of:

\GlsXtrFormatLocationList{⟨*location list*⟩}

This is internally used by \glossaryentrynumbers. The nonumberlist option redefines \glossaryentrynumbers so that it doesn't display the number list, but it still saves the number list in case it's required. The desired font formatting for the location list can now

more easily be set by redefining \GlsXtrFormatLocationList, without interfering with \glossaryentrynumbers.

If you want to suppress the number list always use the nonumberlist option instead of redefining \glossaryentrynumbers to do nothing.

Note that if you are using the "unsrt" family of commands the location list will only be present if the appropriate field has been set (see §8.4.2). There's no need to save locations with bib2gls or with \printnoidxglossary because this is performed automatically (unlike \printglossary where the trick with \glossaryentrynumbers is required to capture the location list).

Sometimes users like to insert "page" or "pages" in front of the location list. This is quite fiddly to do with the base glossaries package, but glossaries-extra provides a way of doing this. First you need to enable this option and specify the text to display using:

```
\GlsXtrEnablePreLocationTag{⟨page tag⟩}{⟨pages tag⟩}
```

where ⟨*page tag*⟩ is the text to display if the location list only contains a single location and ⟨*pages tag*⟩ is the text to display otherwise. For example:

```
\GlsXtrEnablePreLocationTag{Page: }{Pages: }
```

An extra run is required when using this command.

Use glsignore not @gobble as the format if you want to suppress the page number (and only index the entry once or use bib2gls).

See the accompanying sample file sample-pages.tex for an example.

Note that with bib2gls the loc-prefix resource option inserts a prefix at the start of non-empty location lists, which can be used as an alternative to \GlsXtrEnablePreLocationTag. There is also a corresponding loc-suffix option to provide a suffix.

Location lists displayed with \printnoidxglossary internally use \glsnoidxdisplayloc. This command is provided by glossaries, but is modified by glossaries-extra to check for the start and end range formation identifiers (and) which are discarded to obtain the actual control sequence name that forms the location formatting command.

If the range identifiers aren't present, this just uses

```
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
```

otherwise it uses

```
\glsxtrdisplaystartloc{⟨format⟩}{⟨location⟩}
```

for the start of a range (where the identifier has been stripped from ⟨*format*⟩) or

```
\glsxtrdisplayendloc{⟨format⟩}{⟨location⟩}
```

for the end of a range (where the identifier has been stripped from ⟨*format*⟩).

By default the start range command saves the format in:

```
\glsxtrlocrangefmt
```

and does:

```
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
```

(If the format is empty, it will be replaced with `glsnumberformat`.)

The end command checks that the format matches the start of the range, does:

```
\glsxtrdisplayendlochook
```

(which does nothing by default), followed by:

```
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
```

and then sets `\glsxtrlocrangefmt` to empty.

This means that the list

```
\glsnoidxdisplayloc{}{page}{(textbf}{1},
\glsnoidxdisplayloc{}{page}{textbf}{1},
\glsnoidxdisplayloc{}{page}{)textbf}{1}.
```

doesn't display any differently from

```
\glsnoidxdisplayloc{}{page}{textbf}{1},
\glsnoidxdisplayloc{}{page}{textbf}{1},
\glsnoidxdisplayloc{}{page}{textbf}{1}.
```

but it does make it easier to define your own custom list handler that can accommodate the ranges.

8.6.4. Indexing Groups

The letter or symbol or number groups are a by-product of the indexing application. These are usually determined during the sorting according to the first (significant) character of the sort value. If the first character is an alphabetical character, the group is a letter group, with the group label the same as the letter. If the sort value is numeric, the group is a number group, with the label `glsnumbers`, otherwise the group is a symbol group with the label `glssymbols`.

For the "unsrt" family of commands, see §8.4.1 for more details about how group headers are inserted into the glossary. Only those commands are able to support sub-groups.

With `xindy`, the number group is automatically provided with the `xindy=glsnumbers` package option. It can be suppressed with `xindy={glsnumbers=false}` (see the base glossaries user manual for further details).

With `bib2gls`, group formation requires `--group` (or `-g`). This setting is off by default to allow for a faster process where no groups are required. When this setting is on, there are additional groups, depending on the sort method. For example, if you use a date-time sort method, then you will have date-time groups.

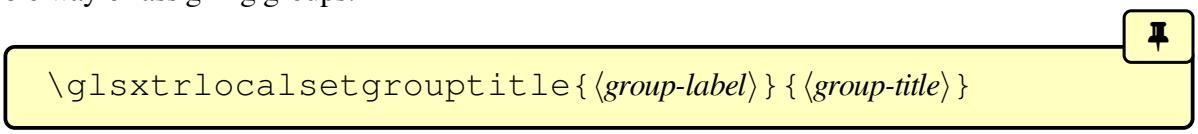
Take care not to confusion groups with hierarchy. See Gallery: Logical Glossary Divisions (type vs group vs parent)[a] for the difference between the `group`, `type` and `parent` fields.

[a] `dickimaw-books.com/gallery/index.php?label=logicaldivisions`

The base glossaries package provides a simplistic way of assigning a title to a group to allow for the use of language-sensitive commands `\glssymbolsgroupname` and `\glsnumbersgroupname`, which correspond to the `glssymbols` and `glsnumbers` groups. The more flexible groups that can be created with `bib2gls` require a better approach that is less likely to cause a conflict.

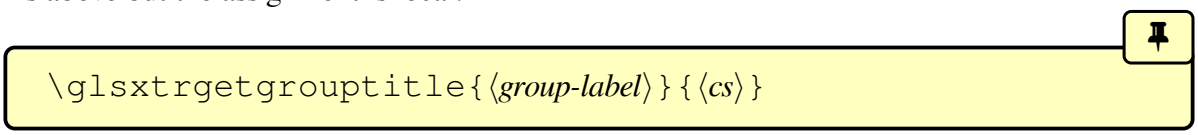
`\glsxtrsetgrouptitle{`⟨*group-label*⟩`}{`⟨*group-title*⟩`}`

Globally assigns the given title ⟨*group-title*⟩ to the group identified by ⟨*group-label*⟩. This command is used implicitly within the `glstex` file to assign titles to groups obtained by `bib2gls`. Judicious definitions of the helper commands provided by `bib2gls` can provide a more flexible way of assigning groups.



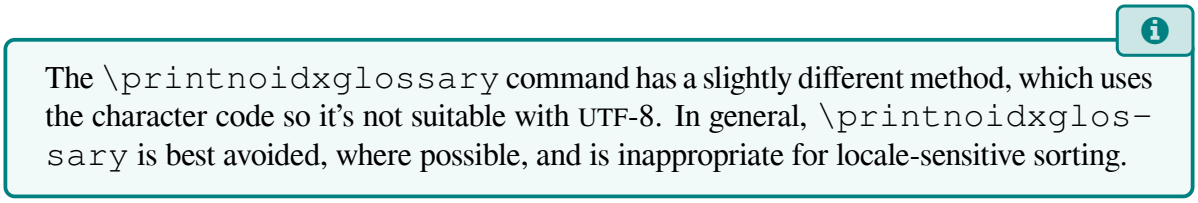
`\glsxtrlocalsetgrouptitle{`⟨*group-label*⟩`}{`⟨*group-title*⟩`}`

As above but the assignment is local.



`\glsxtrgetgrouptitle{`⟨*group-label*⟩`}{`⟨*cs*⟩`}`

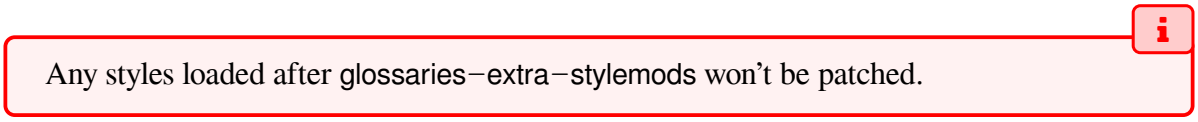
Obtains the title corresponding to the group identified by ⟨*group-label*⟩ and stores the result in ⟨*cs*⟩. This command first checks if a title has been assigned by `\glsxtrgetgrouptitle` and then, for compatibility with the base glossaries package, it will test for the existence of `\`⟨*group-label*⟩`groupname` if ⟨*group-label*⟩ is `glssymbols` or `glsnumbers` or a single character. If no title is obtained from any of these tests, then the title will be assumed to be the same as the label.



The `\printnoidxglossary` command has a slightly different method, which uses the character code so it's not suitable with UTF-8. In general, `\printnoidxglossary` is best avoided, where possible, and is inappropriate for locale-sensitive sorting.

8.6.5. glossaries-extra-stylemods

The glossaries-extra-stylemods package (more conveniently loaded through the glossaries-extra `stylemods` option) modifies some of the predefined styles that are provided with the glossaries package.



Any styles loaded after glossaries-extra-stylemods won't be patched.

The `stylemods` option may be provided without a value, in which case all currently defined styles will be patched. Alternatively, you can supply a comma-separated list as the value, which indicates that, for each ⟨*element*⟩ in the list, the package glossary-⟨*element*⟩ should be loaded and, if it's a package provided with the base glossaries package, patched. For example:

```
\usepackage{glossaries-extra}
\usepackage{glossary-longragged}
\usepackage{glossary-mcols}
\usepackage{glossaries-extra-stylemods}
\setglossarystyle{mcolindex}
```

is equivalent to:

```
\usepackage[stylemods={longragged,mcols},
 style=mcolindex]{glossaries-extra}
```

You may prefer to combine `stylemods` with `nostyles` to reduce the overhead of loading unnecessary packages.

The glossaries-extra-stylemods package adjusts the predefined styles so that they all use `\glspostdescription` and replaces any hard-coded space before the location list with

`\glsxtrprelocation` *initial:* `\space`

You can therefore redefine that command in combination with `postpunc` to alter the separator before the location list. For example, to have a comma followed by `\hfil`:

```
\usepackage[postpunc=comma,stylemods]{glossaries-
extra}
\renewcommand{\glsxtrprelocation}{\hfil}
```

Be careful with doing this as it will look odd if the location list is missing.

With `bib2gls` you can instead redefine `\glsxtrprelocation` to do nothing and set the location prefixes with `loc-prefix` which will only apply if the entry has a location list. Alternatively, you could redefine `\glsxtrprelocation` to check if the `location` field is set.

8.6.5.1. Inline Style

The patched inline style is dealt with slightly differently. The original definition provided by the glossary-inline package uses `\glspostdescription` at the end of the glossary (not after each entry description) within the definition of `\glspostinline`. The style modification changes this so that `\glspostinline` just does a full stop followed by space factor adjustment, and the description `\glsinlinedescformat` and sub-entry description

formats \glsinlinesubdescformat are redefined to include \glsxtrpostdescription (not \glspostdescription). This means that the modified inline style isn't affected by the nopostdot option, but the category post-description hook can still be used.

8.6.5.2. Tabular Styles

The tabular-like styles, such as long are adjusted so that the \ifglsnogroupskip conditional (set with nogroupskip) is moved outside of the definition of \glsgroupskip to avoid problems that cause an "Incomplete \iftrue" error with \printunsrtglossary and \printnoidxglossary. This means that if you want to change this conditional using \setupglossaries or using the nogroupskip option in \printglossary, \printnoidxglossary or \printunsrtglossary, you must also reset the glossary style.

8.6.5.3. List Styles

The list styles use:

```
\glslistprelocation                    initial: \glsxtrprelocation
```

(which defaults to \glsxtrprelocation) for top-level items and:

```
\glslistchildprelocation          initial: \glslistprelocation
```

(which defaults to \glslistprelocation) for child items.

The description (including the post-description hook) is governed by:

```
\glslistdesc{⟨entry-label⟩}
```

for the list and altlist styles (but not the listdotted variations).

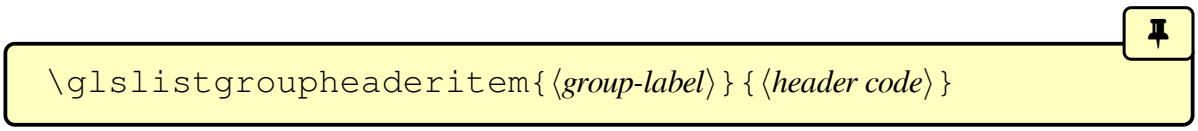
The hard-coded \item[⟨target and name⟩] is replaced with:

```
\glslistitem{⟨entry-label⟩}
```

The altlist styles use:

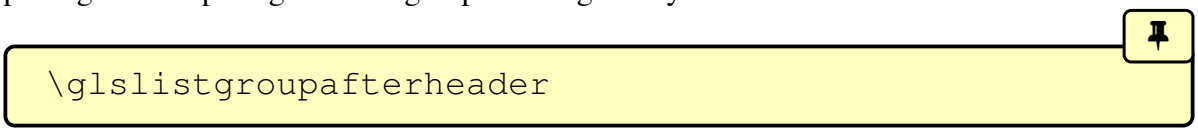
```
\glsaltlistitem{⟨entry-label⟩}
```

which internally uses \glslistitem. The header item (for the list styles that should the group title, such as listgroup) is governed by:



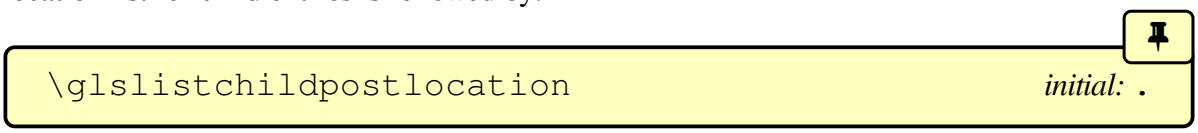
```
\glslistgroupheaderitem{⟨group-label⟩}{⟨header code⟩}
```

This ignores the ⟨*group-label*⟩ by default and simply places the second argument in the optional argument of `\item`. The ⟨*header code*⟩ is the formatted group title, possibly including a hypertarget. The spacing after the group item is given by:



```
\glslistgroupafterheader
```

For just the list style and its letter group variations (not the altlist or listdotted variations) the location list for child entries is followed by:

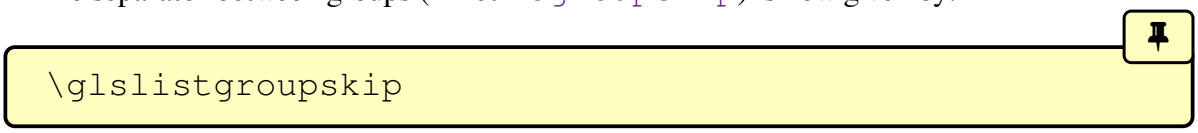


```
\glslistchildpostlocation                    initial: .
```

which defaults to a full stop.

The default value of `\glslistdottedwidth` is changed so that it's set at the start of the document (if it hasn't been changed in the preamble). This should take into account situations where `\hsize` isn't set until the start of the document.

The separator between groups (if not `nogroupskip`) is now given by:

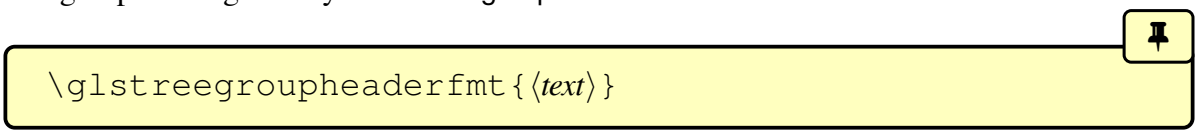


```
\glslistgroupskip
```

This defaults to `\indexspace` with penalties to deter page breaks. This command isn't used if `nogroupskip` is set.

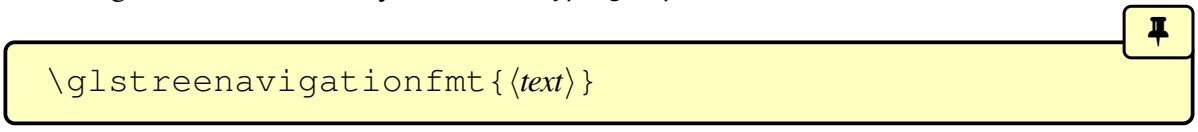
8.6.5.4. Tree Styles

The group headings for styles like treegroup are formatted with:



```
\glstreegroupheaderfmt{⟨text⟩}
```

The navigation elements for styles like treehypergroup is formatted with:



```
\glstreenavigationfmt{⟨text⟩}
```

The above two commands are defined in terms of `\glstreenamefmt`, since that was the command originally used for the group headings and navigation. This now allows these different elements to be defined independently, but the most common redefinition is for `\glstreenamefmt` to remove the bold in the name. If the bold is still required for the group heading and navigation elements, then both other commands also need redefining. To simplify matters, all three commands have been defined to use:

```
\glstreedefaultnamefmt{⟨text⟩}
```

This simply does `\textbf{⟨text⟩}`.

This means that if you want to change all three to use a particular style you only need to redefine `\glstreedefaultnamefmt`, but if you only want to redefine `\glstreenamefmt` without affecting the other two commands, then you now can.

The separator between groups without headers is given by:

```
\glstreegroupskip
```

This defaults to just `\indexspace` without penalties. This command isn't used if `nogroupskip` is set. (The penalties introduced in v1.41 were moved to `\glstreegroupheaderskip` in v1.42 as they are inappropriate when there's no header.)

The separator between groups with headers is now given by:

```
\glstreegroupheaderskip
```

This defaults to `\glstreegroupskip` with penalties to deter page breaks after the group heading.

The styles that display the group titles now use:

```
\glstreePreHeader{⟨group-label⟩}{⟨group-title⟩}
```

This does nothing by default and is inserted before the group title. You can redefine it to add the group title to the PDF bookmarks. For example, if the glossary title uses `\chapter` then:

```
\renewcommand{\glstreePreHeader}[2]{%
 \pdfbookmark[1]{#2}{\currentglossary.#1}%
}
```

will insert section-level bookmarks. The use of `\currentglossary` helps to provide unique bookmark labels in the event of multiple glossaries.

The glossary−tree package provides the commands

```
\glstreepredesc
```

and

```
\glstreechildpredesc
```

(which both default to a space) and uses them in the tree-like styles, but not for the alttree style. The glossaries-extra-stylemods package modifies the alttree style so that it has equivalent hooks:

```
\glsalttreepredesc
```

and

```
\glsalttreechildpredesc
```

These do nothing by default.

The index-like and tree-like styles insert the pre-location list space with:

```
\glstreeprelocation                initial: \glsxtrprelocation
```

(which defaults to `\glsxtrprelocation`) for top-level items and

```
\glstreechildprelocation       initial: \glstreeprelocation
```

(which defaults to `\glstreeprelocation`) for child items.

The styles like treenoname use:

```
\glstreenonamedesc{⟨entry-label⟩}
```

to display the pre-description separator, the description and the post-description hook. Similarly for the symbol:

```
\glstreenonamesymbol{⟨entry-label⟩}
```

The above are just used for top-level entries. Child entries don't have the name or symbol displayed for the treenoname styles, so there's only a command for the child description:

```
\glstreenonamechilddesc{⟨entry-label⟩}
```

For the tree styles (but not the treenoname or alttree styles), the description is displayed using:

```
\glstreedesc{⟨entry-label⟩}
```

and the symbol with:

```
\glstreesymbol{⟨entry-label⟩}
```

Again the above two commands are just for top-level entries. The child entries use:

```
\glstreechilddesc{⟨entry-label⟩}
```

for the description and

```
\glstreechildsymbol{⟨entry-label⟩}
```

for the symbol. There are now wrapper commands for `\glstreedesc` and `\glstreechilddesc` that check for the description and symbol to determine what separator to use before the page list:

```
\glstreeDescLoc{⟨entry-label⟩}{⟨location list⟩}
```

for top-level entries and

```
\glstreeChildDescLoc{⟨entry-label⟩}{⟨location list⟩}
```

for sub-entries.

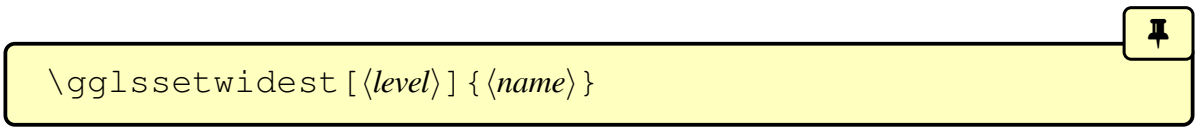
If either the symbol or description is present these will use `\glstreeprelocation` or `\glstreechildprelocation`, respectively. Otherwise, both will use:

```
\glstreeNoDescSymbolPreLocation
```

The default is a space. This means that you could have, say, a comma followed by a space for terms that are simply an alias, but just have a space for terms that have a description that ends with a full stop (or that just have a symbol without a description) where the comma would be inappropriate.

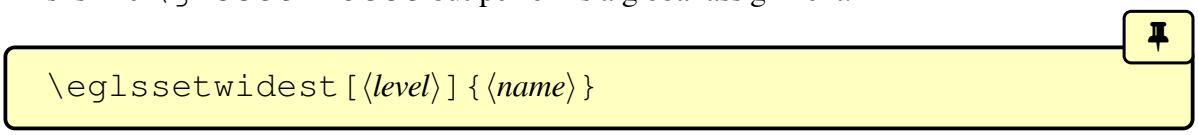
Version 1.42 has corrected an error that was introduced to v1.41 that caused the name to run into the location list if there was no symbol and no description.

There are some additional commands for use with the alttree style to make it easier to modify. These commands are only defined if the glossary−tree package has already been loaded, which is typically the case unless the `notree` or `nostyles` option has been used when loading glossaries.



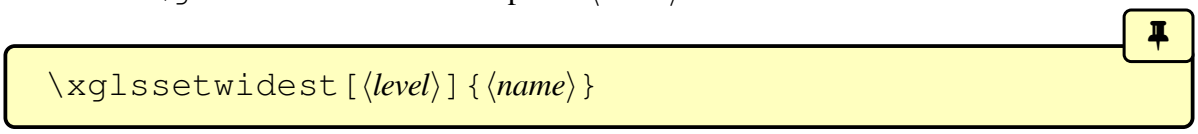
```
\gglssetwidest[⟨level⟩]{⟨name⟩}
```

This is like `\glssetwidest` but performs a global assignment.



```
\eglssetwidest[⟨level⟩]{⟨name⟩}
```

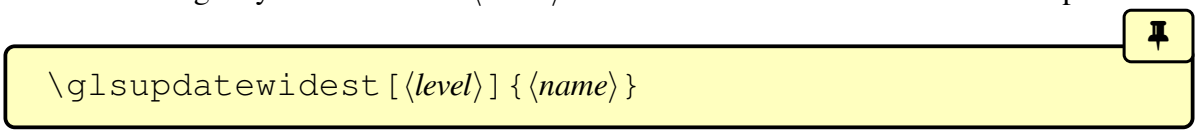
This is like `\glssetwidest` but expands ⟨*name*⟩.



```
\xglssetwidest[⟨level⟩]{⟨name⟩}
```

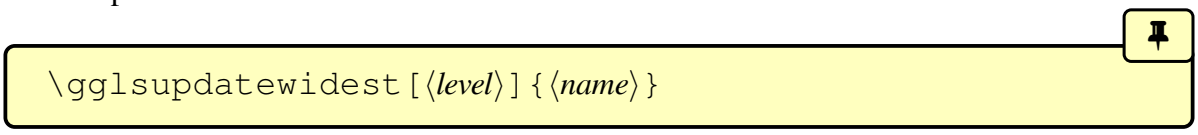
This is like `\eglssetwidest` but performs a global assignment.

The following only set the value if ⟨*name*⟩ is wider than the current value. Local update:



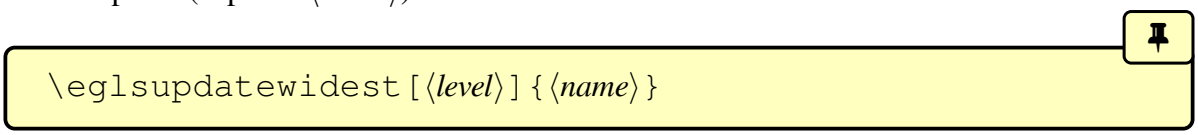
```
\glsupdatewidest[⟨level⟩]{⟨name⟩}
```

Global update:



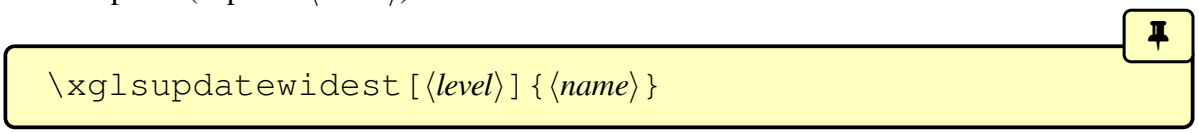
```
\gglsupdatewidest[⟨level⟩]{⟨name⟩}
```

Locale update (expands ⟨*name*⟩):



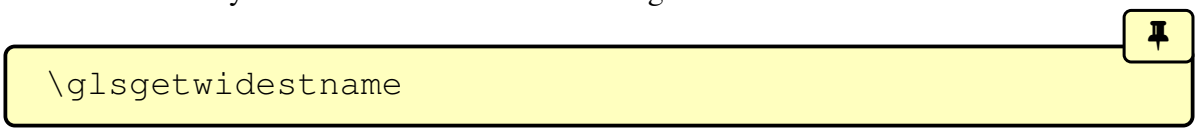
```
\eglsupdatewidest[⟨level⟩]{⟨name⟩}
```

Global update (expands ⟨*name*⟩):



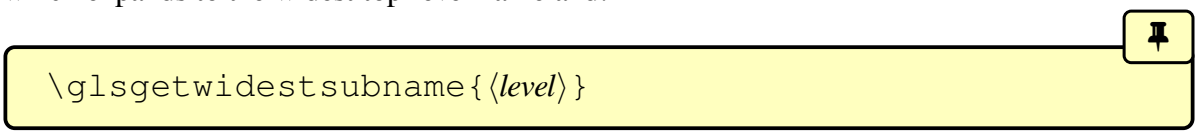
```
\xglsupdatewidest[⟨level⟩]{⟨name⟩}
```

The widest entry value can later be retrieved using:



```
\glsgetwidestname
```

which expands to the widest top-level name and:



```
\glsgetwidestsubname{⟨level⟩}
```

expands to either the widest name for the given hierarchical level or to the widest top-level name, if no widest name set for ⟨*level*⟩.

Note that if you are using bib2gls, you can use the resource option set-widest which will try to determine the widest name of all the selected entries. This isn't guaranteed to work as it may depend on fonts or commands that bib2gls can't replicate, but it should be suitable for names that just consist of text, and can be more efficient than iterating over all the defined entries using TEX.

The command \glsfindwidesttoplevelname provided by glossary-tree has a CamelCase synonym:

```
\glsFindWidestTopLevelName
```

Similar commands are also provided. If the optional ⟨*glossary labels*⟩ is omitted, the list of all non-ignored glossaries is assumed.

```
\glsFindWidestUsedTopLevelName[⟨glossary labels⟩]
```

This has an additional check that the entry has been used. Naturally this is only useful if the glossaries that use the alttree style occur at the end of the document. This command should be placed just before the start of the glossary. (Alternatively, place it at the end of the document and save the value in the auxiliary file for the next run.)

```
\glsFindWidestUsedAnyName[⟨glossary labels⟩]
```

This is like the previous command but if doesn't check the parent key. This is useful if all hierarchical levels should have the same width for the name.

```
\glsFindWidestAnyName[⟨glossary labels⟩]
```

This is like the previous command but doesn't check if the entry has been used.

```
\glsFindWidestUsedLevelTwo[⟨glossary labels⟩]
```

This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```
\glsFindWidestLevelTwo[⟨glossary labels⟩]
```

This is like the previous command but doesn't check if the entry has been used.

```
\glsFindWidestUsedAnyNameSymbol[⟨glossary labels⟩]{⟨register⟩}
```

This is like \glsFindWidestUsedAnyName but also measures the symbol. The length of the widest symbol is stored in ⟨*register*⟩.

\glsFindWidestAnyNameSymbol[⟨*glossary labels*⟩]{⟨*register*⟩}

This is like the previous command but it doesn't check if the entry has been used.

\glsFindWidestUsedAnyNameSymbolLocation[⟨*glossary labels*⟩]{⟨*register1*⟩}{⟨*register2*⟩}

This is like \glsFindWidestUsedAnyNameSymbol but also measures the location list. This requires \glsentrynumberlist. The length of the widest symbol is stored in ⟨*register1*⟩ and the length of the widest location list is stored in ⟨*register2*⟩.

\glsFindWidestAnyNameSymbolLocation[⟨*glossary labels*⟩]{⟨*register1*⟩}{⟨*register2*⟩}

This is like the previous command but it doesn't check if the entry has been used.

\glsFindWidestUsedAnyNameLocation[⟨*glossary labels*⟩]{⟨*register*⟩}

This is like \glsFindWidestUsedAnyNameSymbolLocation but doesn't measure the symbol. The length of the widest location list is stored in ⟨*register*⟩.

\glsFindWidestAnyNameLocation[⟨*glossary labels*⟩]{⟨*register*⟩}

This is like the previous command but doesn't check if the entry has been used.

The layout of the symbol, description and location list is governed by:

\glsxtralttreeSymbolDescLocation{⟨*entry-label*⟩}{⟨*location list*⟩}

for top-level entries and

\glsxtralttreeSubSymbolDescLocation{⟨*entry-label*⟩}{⟨*location list*⟩}

for sub-entries.

There is now a user level command that performs the initialisation for the alttree style:

```
\glsxtralttreeInit
```

The paragraph indent for subsequent paragraphs in multi-paragraph descriptions is provided by the length:

```
\glsxtrAltTreeIndent
```

For additional commands that are available with the alttree style, see the documented code (`glossaries-extra-code.pdf`). See also the accompanying sample files `sample-alttree.tex`, `sample-alttree-sym.tex` and `sample-alttree-marginpar.tex`.

8.7. New Glossary Styles

The glossaries-extra package comes with some new styles. The associated style package needs to be loaded. This can be done with `\usepackage` but it's simpler to use the `stylemods` option. For example:

```
\usepackage[stylemods=bookindex,style=bookindex]
{glossaries-extra}
```

If you don't require any of the base styles, use `nostyles` (but note that some style packages automatically load another style package if it the style builds on an existing one).

8.7.1. glossary-bookindex package

The glossary-bookindex package provides the glossary style bookindex. This is very similar to the mcolindexgroup style but is designed for indexes, so by default only the name and location list are displayed. This style is demonstrated in Example 147 (using `bib2gls`). Note that some entries don't have location lists because they weren't recorded in the document, but were included as dependencies. See §5.9.1 for dealing with cross-references that may not be required.

The bookindex style only supports a maximum hierarchical level of 2 (top-level, level 1 and level 2). It's primarily designed for use with `bib2gls`. It may be used with other indexing options, but some features may not be present and UTF-8 characters may cause a problem with non-Unicode engines in letter group headings or PDF bookmarks. (`bib2gls` uses numeric identifies by default to avoid these problems, see §8.4.1.)

The number of columns is given by:

Example 147: The `bookindex` style

Glossary

A

adipiscing, 1
alias-lorem, *see* lorem
amet, 1, *see* dolor
arcu, 1–3, *see also* placerat, vitae & curabitur
augue, 2

C

curabitur

D

dolor
donec, 2

E

et, *see* vulputate

L

lectus
  urna, *see* nulla
lorem, 1, 3

M

mauris

N

nonummy, *see also* mauris
  consectetuer, 2
  vulputate, 1
nulla, 3

P

pellentesque, *see* augue
placerat, 1

V

vitae

\glsxtrbookindexcols *initial:* 2

which defaults to 2.

This style uses the multicols environment. If the command:

\glsxtrbookindexcolspread

isn't empty then it's supplied as the optional argument following \begin{multicols} {⟨*n*⟩}. You can switch from multicols to multicols* by redefining:

\glsxtrbookindexmulticolsenv

For example:

```
\renewcommand{\glsxtrbookindexmulticolsenv}
{multicols*}
```

The target for top-level items is created with:

\glsxtrbookindextarget{⟨*entry-label*⟩}{⟨*text*⟩}

This simply uses \glstarget but may be redefined to insert additional content in front of the target or to suppress the target according to some condition (such as the entry's category).

Remember that if you want to use \glsxtrtarget then *all* instances of \glstarget for the entire document (including any standalone entries or other glossaries with different styles) need to be replaced with that command in order for it to work correctly. The simplest way to do this is to redefine \glstarget at the start.

The target for child items is created with:

\glsxtrbookindexsubtarget{⟨*entry-label*⟩}{⟨*text*⟩}

This simply uses \glsxtrbookindextarget.

Each top-level entry is displayed using:

\glsxtrbookindexname{⟨*entry-label*⟩}

This just does \glossentryname{⟨entry-label⟩} by default. For example, if you want the symbol to be included:

```
\renewcommand*{\glsxtrbookindexname}[1]{%
 \glossentryname{#1}%
 \ifglshassymbol{#1}
  {\space (\glossentrysymbol{#1})}{}%
}
```

or if you want the description (if set):

```
\renewcommand*{\glsxtrbookindexname}[1]{%
 \glossentryname{#1}%
 \ifglshasdesc{#1}
  {\space \glossentrydesc{#1}\glspostdescription}%
}
```

(which picks up the post-description hook).

Alternatively you can use the \glsxtrpostname⟨category⟩ hook to append information after the name according to the entry's category.

Sub-entries are displayed using:

```
\glsxtrbookindexsubname{⟨entry-label⟩}
```

which just defaults to \glsxtrbookindexname{⟨entry-label⟩}.

The separator used before the location list for top-level entries is given by:

```
\glsxtrbookindexprelocation{⟨entry-label⟩}
```

where ⟨entry-label⟩ is the entry's label. This checks if the location field has been set. If it has, it does:

```
,\glsxtrprelocation
```

otherwise it just does \glsxtrprelocation (which defaults to \space) with no comma. If you're using bib2gls with save-locations=false, the location field won't be set.

The separator used before the location list for sub-entries is given by:

```
\glsxtrbookindexsubprelocation{⟨entry-label⟩}
```

which defaults to `\glsxtrbookindexprelocation{entry-label}`.

The actual location list is encapsulated with:

```
\glsxtrbookindexlocation{⟨entry-label⟩}{⟨location list⟩}
```

for top-level entries and:

```
\glsxtrbookindexsublocation{⟨entry-label⟩}{⟨location list⟩}
```

for sub-entries. These both just do ⟨*location list*⟩ by default.

The separator used between a top-level parent and child entry is given by:

```
\glsxtrbookindexparentchildsep
```

This defaults to `\nopagebreak`.

The separator used between a sub-level parent and child entry is given by:

```
\glsxtrbookindexparentsubchildsep
```

This defaults to `\glsxtrbookindexparentchildsep`.

The separator between top-level entries is given by:

```
\glsxtrbookindexbetween{⟨entry1-label⟩}{⟨entry2-label⟩}
```

This comes after the entry given by ⟨*entry1-label*⟩, if the entry has no children, or after the last descendent otherwise, so it always comes immediately before the entry given by ⟨*entry2-label*⟩ unless the entry occurs at the start of a group. This does nothing by default.

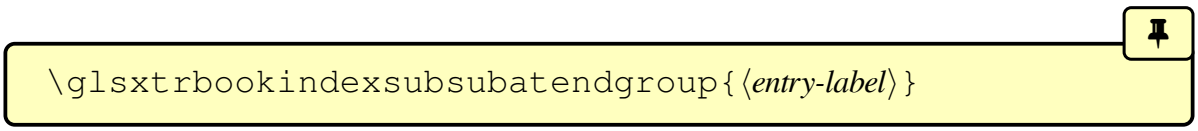
The separator between two level 1 entries is given by:

```
\glsxtrbookindexsubbetween{⟨entry1-label⟩}{⟨entry2-label⟩}
```

The separator between two level 2 entries is given by:

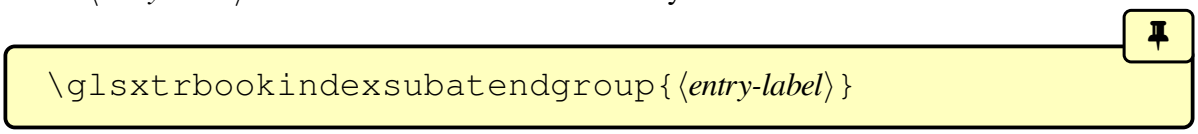
```
\glsxtrbookindexsubsubbetween{⟨entry1-label⟩}{⟨entry2-label⟩}
```

At the end of each letter group, the following hooks are done in order:



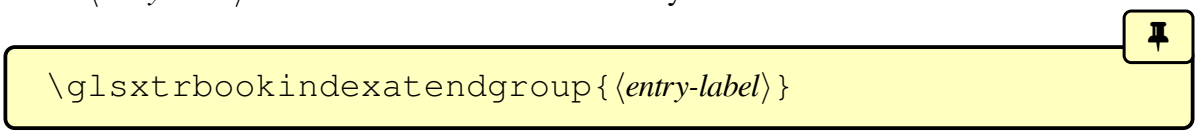
\glsxtrbookindexsubsubatendgroup{⟨*entry-label*⟩}

where ⟨*entry-label*⟩ is the label of the last level 2 entry



\glsxtrbookindexsubatendgroup{⟨*entry-label*⟩}

where ⟨*entry-label*⟩ is the label of the last level 1 entry



\glsxtrbookindexatendgroup{⟨*entry-label*⟩}

where ⟨*entry-label*⟩ is the label of the last level 0 entry.

For example, the resource option seealso=omit instructs bib2gls to omit the seealso cross-reference from the location list. (The see cross-reference will still be added unless you also have see=omit.) The seealso cross-reference can instead be appended after the child entries using:

```
\renewcommand{\glsxtrbookindexatendgroup}[1]{%
  \glsxtrifhasfield{seealso}{#1}%
  {\glstreesubitem\glsxtruseseealso{#1}}{}%
}
\renewcommand{\glsxtrbookindexbetween}[2]{%
  \glsxtrbookindexatendgroup{#1}%
}

\renewcommand{\glsxtrbookindexsubatendgroup}[1]{%
  \glsxtrifhasfield{seealso}{#1}%
  {\glstreesubsubitem\glsxtruseseealso{#1}}{}%
}

\renewcommand{\glsxtrbookindexsubbetween}[2]{%
  \glsxtrbookindexsubatendgroup{#1}%
}

\renewcommand{\glsxtrbookindexsubsubatendgroup}[1]
{%
  \glsxtrifhasfield{seealso}{#1}%
  {\glstreeitem\hspace*{40pt}\glsxtruseseealso{#1}}
{}%
}
```

```
\renewcommand{\glsxtrbookindexsubsubbetween}[2]{%
  \glsxtrbookindexsubsubatendgroup{#1}%
}
```

This uses \glstreesubitem and \glstreesubsubitem to indent the cross-reference according to the next level down, so the cross-reference for a top-level entry is aligned with the sub-entries, and a level 1 entry has its cross-reference aligned with sub-sub-entries. In the event that a level 2 entry has a cross-reference, this is indented a bit further (but it won't be aligned with any deeper level as the bookindex style only supports a maximum of two sub-levels).

As from version 1.54, the bookindex style uses \glstreesubsubitem indirectly via:

```
\glsxtrbookindexsubsubitem{⟨level⟩}
```

The argument ⟨*level*⟩ will be 2 or more. This simply expands to \glstreesubsubitem, ignoring its argument, but may be redefined if required.

The bookindex style uses group headings. (If you use bib2gls remember to invoke it with the --group or -g switch, see §8.4.1.) The heading will use:

```
\glsxtrbookindexbookmark{⟨group-title⟩}{⟨bookmark-name⟩}
```

If \pdfbookmark has been defined, this will use that command to bookmark the group title. If section=chapter is set (default if chapters are defined) then this uses level 1 otherwise it uses level 2. You can redefine this command if this isn't appropriate. If \pdfbookmark hasn't been defined, this command does nothing.

The group heading is formatted according to:

```
\glsxtrbookindexformatheader{⟨group-title⟩}
```

which is defined as:

```
\newcommand*{\glsxtrbookindexformatheader}[1]{%
  \par{\centering\glstreegroupheaderfmt{#1}\par}%
}
```

where \glstreegroupheaderfmt is provided by the glossary−tree package, which is automatically loaded. Note that the entry names aren't encapsulated with \glstreenamefmt.

The skip after a group header is given by:

```
\glsxtrbookindexpregroupskip{⟨skip⟩}
```

The argument is the skip that would normally be inserted if there wasn't a group header.

The glossary−bookindex package provides some supplementary commands that aren't used by default, but may be used when adjusting the style. These commands should only be used within one of the \print...glossary commands. (That is, they should only be used in glossary styles or in hooks.)

```
\glsxtrbookindexmarkentry{⟨entry-label⟩}
```

This writes information to the aux file that can be read on the next run to obtain the first and last entry on each page of the glossary.

You can display the first entry associated with the current page using:

```
\glsxtrbookindexfirstmark
```

and the last entry associated with the current page using:

```
\glsxtrbookindexlastmark
```

These do nothing if there are no entries marked on the current page (or if the document build isn't up to date).

The entry is formatted using:

```
\glsxtrbookindexfirstmarkfmt{⟨entry-label⟩}
```

for the first instance and

```
\glsxtrbookindexlastmarkfmt{⟨entry-label⟩}
```

for the last.

These commands are designed for use in page headers or footers where the page number is stable. For example, \glsxtrbookindexname can be redefined to mark the current entry:

```
\renewcommand{\glsxtrbookindexname}[1]{%
  \glsxtrbookindexmarkentry{#1}%
  \glossentryname{#1}%
}
```

If you only want to mark the top-level entries, remember to redefine \glsxtrbookindex-subname as it defaults to \glsxtrbookindexname:

```
\renewcommand{\glsxtrbookindexsubname}[1]{%
  \glossentryname{#1}%
}
```

Then if you're using fancyhdr you can set the page style to show the first and last entry for the current page with:

```
\pagestyle{fancy}%
\lhead{\thepage}%
\lfoot{\glsxtrbookindexfirstmark}%
\cfoot{}%
\rfoot{\glsxtrbookindexlastmark}%
```

8.7.2. glossary−longextra package

The glossary−longextra package provides additional tabular-like styles similar to those provided by glossary−longbooktabs (which is automatically loaded). These don't support hierarchical levels except for homographs (level 1 entries with the same name as their parent).

By default, these styles use the longtable environment, but if you know that your glossary won't span more than a page and you need to use it in a context that's incompatible with longtable, you can instead setup these styles to use tabular instead. In order to do this you must use:

```
\GlsLongExtraUseTabulartrue
```

before the style is set. If you later want to switch back to using longtable for another glossary, use:

```
\GlsLongExtraUseTabularfalse
```

(or scope `\GlsLongExtraUseTabulartrue`). Again, the style must be set after this change to the conditional is implemented. You can test this setting with:

```
\ifGlsLongExtraUseTabular ⟨true⟩\else ⟨false⟩\fi
                                        initial: \iffalse
```

For example:

```
\GlsLongExtraUseTabulartrue
\setglossarystyle{long-name-desc}
```

or

```
\GlsLongExtraUseTabulartrue
\printunsrtglossary[style=long-name-desc]
```

If you switch to tabular, the default vertical alignment is obtained from:

```
\glslongextraTabularVAlign                    initial: c
```

This should expand to one of: `c` (centred), `t` (top) or `b` (bottom). The default is `c`.

For either tabular or longtable, the column titles are formatted according to:

```
\glslongextraHeaderFmt{⟨text⟩}
```

which simply does `\textbf{⟨text⟩}` by default. As with the long-like styles, the header text for the columns are given by the language-sensitive commands: `\entryname`, `\descriptionname`, `\symbolname` and `\pagelistname`.

Most styles show the `name` which, as with other predefined styles, also includes the entry item number (if `entrycounter` is on) and hypertarget anchor. These are all performed for top-level entries with:

```
\glslongextraNameFmt{⟨entry-label⟩}
```

This uses `\glossentryname`, so it supports the post-name hook and associated attributes. Child entries are displayed with:

```
\glslongextraSubNameFmt{⟨level⟩}{⟨entry-label⟩}
```

This includes the sub-entry item number (if `subentrycounter` is on) and the hypertarget anchor. The actual name isn't shown by default.

The horizontal alignment for the name column is obtained with:

```
\glslongextraNameAlign                        initial: l
```

This expands to `l` by default.

For styles that show the `description`, that's formatted with:

```
\glslongextraDescFmt{⟨entry-label⟩}
```

for top-level entries, which uses `\glossentrydesc` and the post-description hook, and

```
\glslongextraSubDescFmt{⟨level⟩}{⟨entry-label⟩}
```

for child entries (which just uses `\glslongextraDescFmt`).

The horizontal alignment for the description column is obtained with:

```
\glslongextraDescAlign
```

This expands to `>{\raggedright}p{\glsdescwidth}` by default. This means ragged-right paragraph style with width given by `\glsdescwidth`. (See the documentation for the array package for information about this alignment syntax.) If a widest name has been set, `\glsdescwidth` will be calculated according to the best fit for the given style.

If you are using `bib2gls`, you may be able to use the `set-widest` option, otherwise to set the widest name, use:

```
\glslongextraSetWidest{⟨widest-name⟩}
```

If you have already used `\glssetwidest` provided with the alttree style, the default widest name will be obtained from that, but note that only level 0 is supported for the glossary-longextra styles.

You can update the widest name with:

```
\glslongextraUpdateWidest{⟨name⟩}
```

This is like `\glslongextraSetWidest` but will only set the new value if it's wider than the current widest name.

Although these styles don't support hierarchy, the following is provided for child entries:

```
\glslongextraUpdateWidestChild{⟨level⟩}{⟨name⟩}
```

This does nothing by default. If `\glslongextraSubNameFmt` is redefined to show the child name, then the above command will need to be redefined to use `\glslongextraUpdateWidest`.

For styles that show the location list, that's formatted with:

```
\glslongextraLocationFmt{⟨entry-label⟩}{⟨location list⟩}
```

for top-level entries. Child location lists are formatted with:

```
\glslongextraSubLocationFmt{⟨level⟩}{⟨entry-label⟩}{⟨location list⟩}
```

Both of these simply do the ⟨*location list*⟩ argument.

The horizontal alignment for the location list column is obtained with:

```
\glslongextraLocationAlign
```

This expands to `>{\raggedright}p{\glspagelistwidth}` by default. This means ragged-right paragraph style with width given by `\glspagelistwidth`.

For styles that show the `symbol` (in addition to the `name`), that's formatted with:

```
\glslongextraSymbolFmt{⟨entry-label⟩}
```

for top-level entries. This simply uses `\glossentrysymbol`. Child entries use:

```
\glslongextraSubSymbolFmt{⟨level⟩}{⟨entry-label⟩}
```

which uses `\glslongextraSymbolFmt`.

The horizontal alignment for the symbol column (except for the long−sym−desc and long−desc−sym styles) is obtained with:

```
\glslongextraSymbolAlign                                    initial: c
```

This expands to `c` by default.

Top-level group headings are formatted with:

```
\glslongextraGroupHeading{⟨number columns⟩}{⟨group-label⟩}
```

The first argument is the total number of columns in the table. For example, 2 for the long−name−desc style or 3 for the long−name−sym−desc style. The second argument is the group's label (not the title). This command does nothing by default. (If you are using `bib2gls`, remember that you need to use the `--group` or `-g` switch to support groups.)

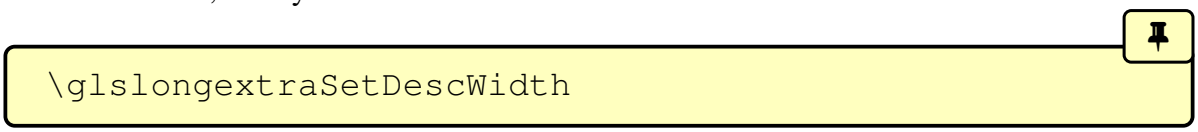
Sub-level groups are only supported with the "unsrt" family of commands (see §8.4.1). When they are supported, the heading will be formatted with:

```
\glslongextraSubGroupHeading{⟨number columns⟩}{⟨prev group level⟩}{⟨group level⟩}{⟨parent-entry-label⟩}{⟨group-label⟩}
```

The styles are sub-divided below into the set of elements that are shown in each column, which may consist of: name, symbol, description or location list. There will be blank cells if any of the corresponding fields have not been set or if the location list has been suppressed.

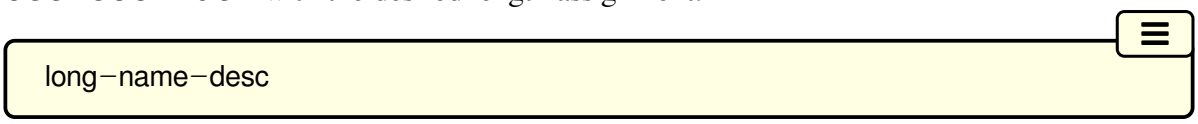
8.7.2.1. Name and Description Only

These styles don't display the symbol or location list, regardless of whether or not they have been set. In each case, the style starts with:

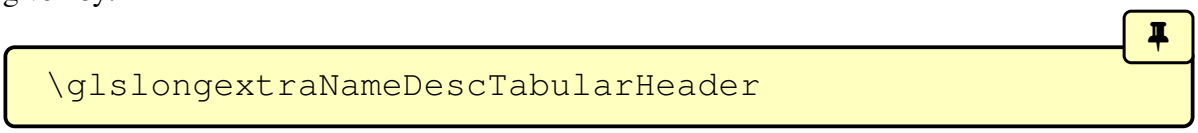


which updates \glsdescwidth according to the widest name, identified with \glslongextraSetWidest. The column header text is also taken into account. If a widest name hasn't been set and the column header is shorter than one or more names, the description column may be too wide. The value of \glsdescwidth is calculated as \linewidth − 4\tabcolsep − W, where W is the width of the widest name.

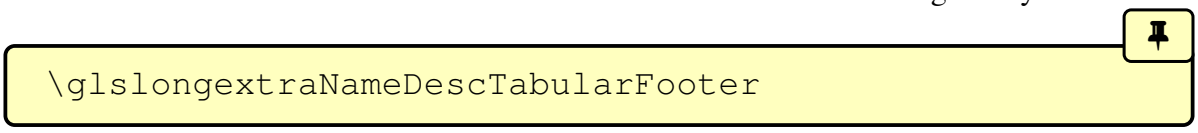
If you want to set \glsdescwidth to a specific value, then redefine \glslongextraSetDescWidth with the desired length assignment.



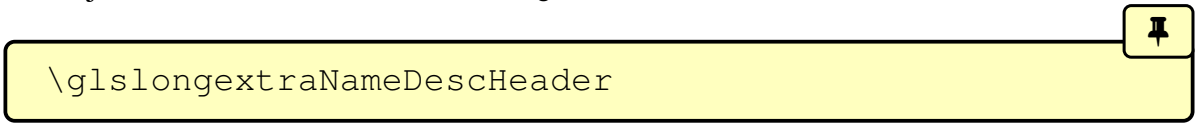
This has two columns: the name on the left and the description on the right. The table header is given by:



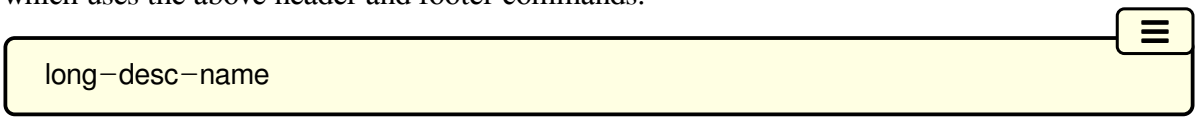
which shows the column headers with horizontal rules. The table footer is given by:



which just does a horizontal rule. With longtable, the table header and footer are set with:



which uses the above header and footer commands.



This has two columns: the name on the right and the description on the left. The table header is given by:

```
\glslongextraDescNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraDescNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraDescNameHeader
```

which uses the above header and footer commands.

8.7.2.2. Name, Symbol and Description Only

These styles don't show the location list. In each case, the style starts with:

```
\glslongextraSymSetDescWidth
```

which updates `\glsdescwidth` according to the widest name, identified with `\glslongextraSetWidest`. This starts by calculating `\glsdescwidth` with `\glslongextraSetDescWidth` and then subtracts the width of the symbol column header text (which is assumed to be the widest text in that column).

If you want to set `\glsdescwidth` to a specific value, then redefine `\glslongextraSymSetDescWidth` with the desired length assignment.

```
long-name-desc-sym
```

This has three columns: the name on the left, the description in the middle and the symbol on the right. The table header is given by:

```
\glslongextraNameDescSymTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraNameDescSymTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraNameDescSymHeader
```

which uses the above header and footer commands.

long-name-sym-desc

This has three columns: the name on the left, the symbol in the middle and the description on the right. The table header is given by:

```
\glslongextraNameSymDescTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraNameSymDescTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraNameSymDescHeader
```

which uses the above header and footer commands.

long-sym-desc-name

This has three columns: the name on the right, the description in the middle and the symbol on the left. The table header is given by:

```
\glslongextraSymDescNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraSymDescNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraSymDescNameHeader
```

which uses the above header and footer commands.

long-desc-sym-name

This has three columns: the name on the right, the symbol in the middle and the description on the left. The table header is given by:

```
\glslongextraDescSymNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraDescSymNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraDescSymNameHeader
```

which uses the above header and footer commands.

8.7.2.3. Name, Description and Location Only

These styles don't display the symbol, regardless of whether or not the `symbol` field has been set. In each case, the style starts with:

```
\glslongextraLocSetDescWidth
```

which updates `\glsdescwidth` according to the widest name, identified with `\glslongextraSetWidest`. This starts by calculating `\glsdescwidth` with `\glslongextraSetDescWidth` and then subtracts 2`\tabcolsep`−`\glspagelistwidth`.

If you want to set `\glsdescwidth` to a specific value, then redefine `\glslongextraLocSetDescWidth` with the desired length assignment.

```
long-name-desc-loc
```

This has three columns: the name on the left, the description in the middle and the location list on the right. The table header is given by:

```
\glslongextraNameDescLocationTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraNameDescLocationTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraNameDescLocationHeader
```

which uses the above header and footer commands.

long-loc-desc-name

This has three columns: the name on the right, the description in the middle and the location list on the left. The table header is given by:

```
\glslongextraLocationDescNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraLocationDescNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraLocationDescNameHeader
```

which uses the above header and footer commands.

8.7.2.4. Name, Description, Symbol and Location

These styles show the name, description, symbol and location list. In each case, the style starts with:

```
\glslongextraSymLocSetDescWidth
```

which updates `\glsdescwidth` according to the widest name, identified with `\glslongextraSetWidest`. This starts by calculating `\glsdescwidth` with `\glslongextraSymSetDescWidth` and then subtracts `2\tabcolsep-\glspagelistwidth`.

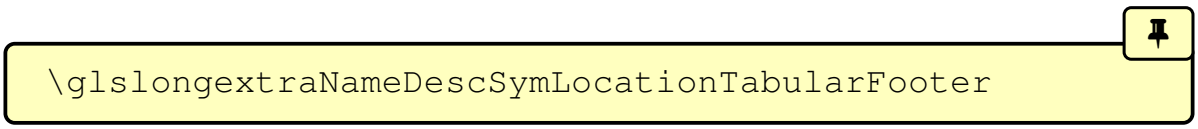
If you want to set `\glsdescwidth` to a specific value, then redefine `\glslongextraSymLocSetDescWidth` with the desired length assignment.

long-name-desc-sym-loc

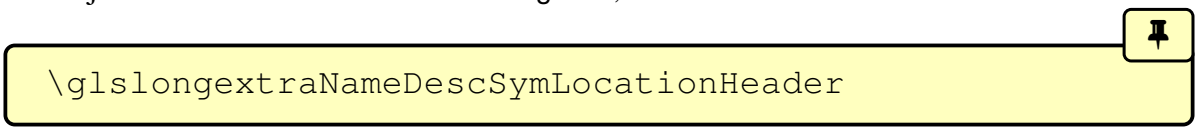
This has four columns, from left to right: the name, description, symbol and the location list. The table header is given by:

```
\glslongextraNameDescSymLocationTabularHeader
```

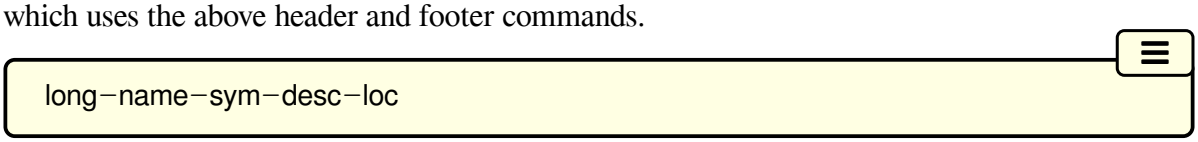
which shows the column headers with horizontal rules. The table footer is given by:



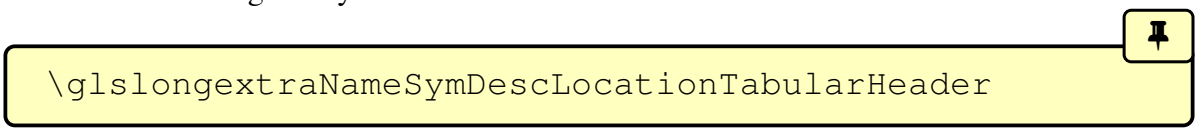
which just does a horizontal rule. With longtable, the table header and footer are set with:



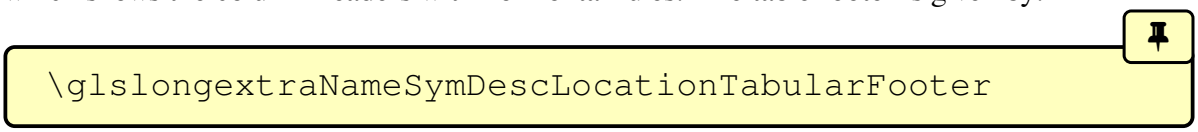
which uses the above header and footer commands.



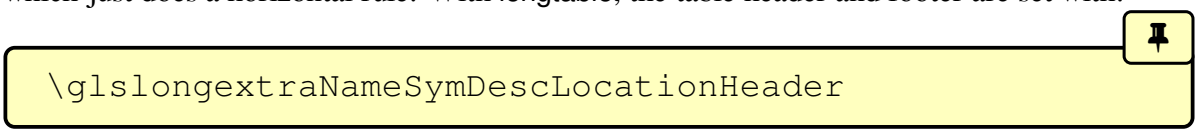
This has four columns, from left to right: the name, symbol, description and the location list. The table header is given by:



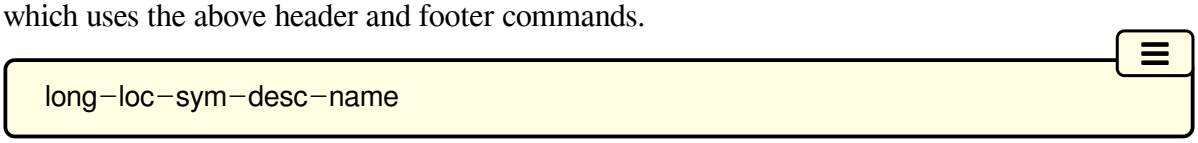
which shows the column headers with horizontal rules. The table footer is given by:



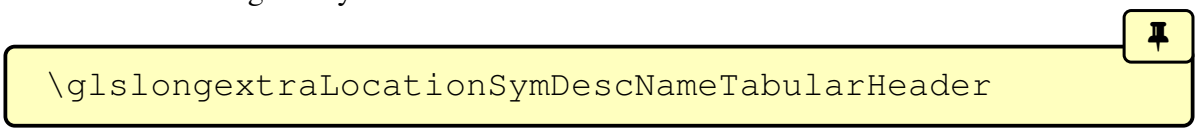
which just does a horizontal rule. With longtable, the table header and footer are set with:



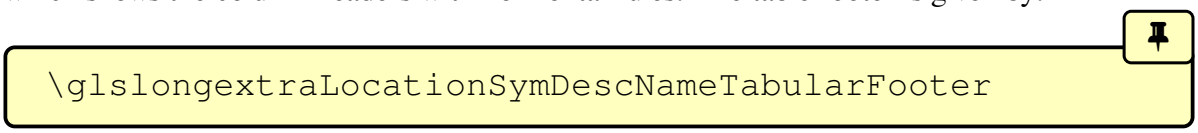
which uses the above header and footer commands.



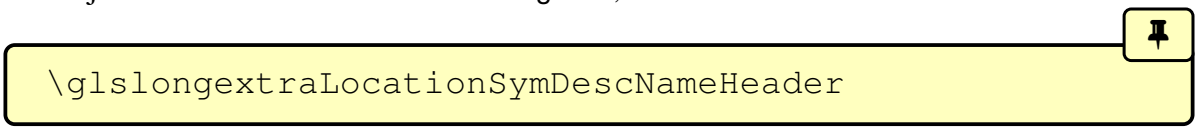
This has four columns, from left to right: the location list, symbol, description and the name. The table header is given by:



which shows the column headers with horizontal rules. The table footer is given by:



which just does a horizontal rule. With longtable, the table header and footer are set with:



which uses the above header and footer commands.

long-loc-desc-sym-name

This has four columns, from left to right: the location list, description, symbol and the name. The table header is given by:

\glslongextraLocationDescSymNameTabularHeader

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraLocationDescSymNameTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraLocationDescSymNameHeader

which uses the above header and footer commands.

8.7.2.5. Symbol and Description Only

These are two-column styles designed to show only the symbol and description. However, if the `symbol` isn't set then the name will be used instead. If this occurs, you may need to change the width of the description column.

The horizontal alignment for the symbol column is obtained with:

\glslongextraSymbolNameAlign *initial:* `l`

which expands to `l` by default. Note that this is different from the alignment used for styles like long-name-sym-desc.

These styles have the entry item number (if `entrycounter` is on) and the hypertarget anchor (if enabled) in the symbol column since there's no name shown (unless the symbol is missing). These are all performed by for top-level entries by:

\glslongextraSymbolTargetFmt{⟨*entry-label*⟩}

The symbol is formatted according to `\glslongextraSymbolFmt`. Child entries use:

\glslongextraSubSymbolTargetFmt{⟨*level*⟩}{⟨*entry-label*⟩}

Unlike `\glslongextraSubNameFmt` this shows the field value (formatted with `\glslongextraSymbolFmt`).

The following commands use the above if the symbol field is set, otherwise they show the name.

```
\glslongextraSymbolOrName{⟨entry-label⟩}
```

Shows the symbol, if set, or the name otherwise, with the target. Child entries use:

```
\glslongextraSubSymbolOrName{⟨level⟩}{⟨entry-label⟩}
```

Shows the symbol with \glslongextraSubSymbolTargetFmt, if set, or the name otherwise, with the target.

In each case, the style starts with:

```
\glslongextraSymNoNameSetDescWidth
```

which calculates \glsdescwidth as \linewidth − 4\tabcolsep − W, where W is the width of the symbol column header. Note that this assumes the content of the symbol column isn't wider than the column header.

If you want to set \glsdescwidth to a specific value, then redefine \glslongextraSymNoNameSetDescWidth with the desired length assignment. For example, if you have a mixture of entries with symbols and some without, which means that there will be a name shown that's wider than the symbol column header, then set the widest name (for example, with the set-widest resource option) and add the following redefinition:

```
\renewcommand{\glslongextraSymNoNameSetDescWidth}{%
 \glslongextraSetDescWidth
}
```

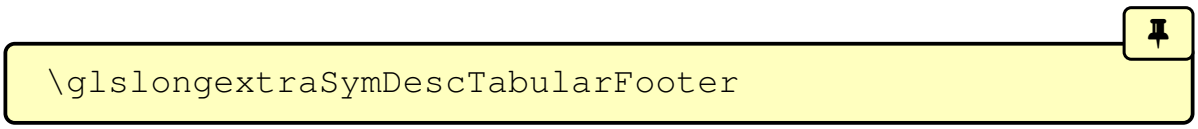
Note that, in this case, if you don't set the widest name then the description column will end up even wider (and therefore cause the table to be even wider) if the name header is narrower than the symbol header.

```
long-sym-desc
```

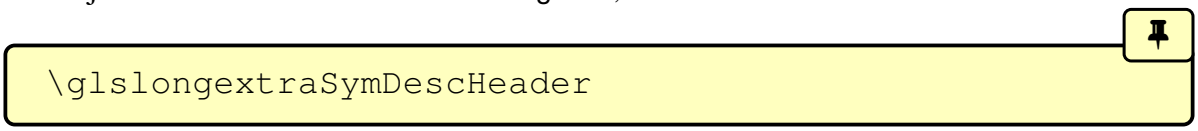
The symbol is in the left column (or the name, if the symbol isn't set). The description is in the right. The location list isn't shown. The table header is given by:

```
\glslongextraSymDescTabularHeader
```

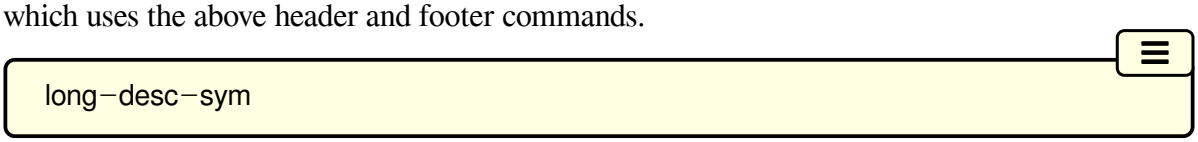
which shows the column headers with horizontal rules. The table footer is given by:



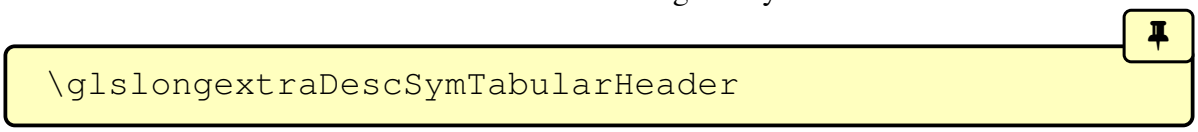
which just does a horizontal rule. With longtable, the table header and footer are set with:



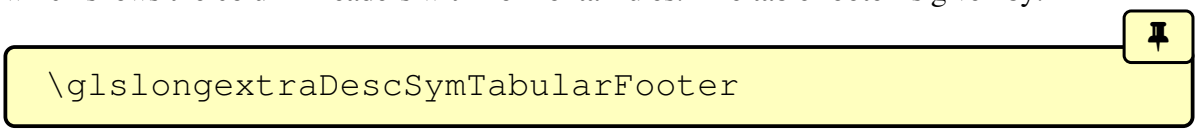
which uses the above header and footer commands.



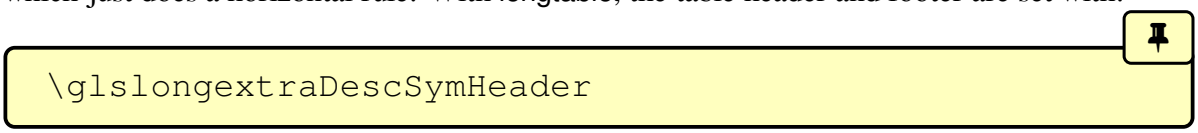
The symbol is in the right column (or the name, if the symbol isn't set). The description is in the left. The location list isn't shown. The table header is given by:



which shows the column headers with horizontal rules. The table footer is given by:



which just does a horizontal rule. With longtable, the table header and footer are set with:



which uses the above header and footer commands.

8.7.2.6. Abbreviations Only

These styles are designed for abbreviations. They display the short and long forms, rather than the name and description, although these may happen to match. They are primarily intended for mini-glossaries or similar summary lists.

Although these styles don't show the name or description, they still use some of the name and description settings provided by glossary-longextra. The column for the short form uses the same alignment as for the name columns (`\glslongextraNameAlign`). The column for the long form uses the same alignment as for the description columns (`\glslongextraDescAlign`) and has the width set to `\glsdescwidth`. However, the name and description formatting commands or attributes (such as `\glsnamefont`, `glossnamefont` or `glossname`) aren't used as the formatting is left to the abbreviation style.

If the `short` field hasn't been set, the short column will show the name instead, and if the `long` field hasn't been set, the long column will show the description instead (using the same commands as for styles like long-name-desc, which do use the associated formatting commands and attributes).

These styles use the following commands:

\glslongextraShortHeader *initial:* \entryname

The header for the column showing the short form. This is defined as:

```
\newcommand{\glslongextraShortHeader}{\entryname}
```

\glslongextraLongHeader *initial:* \descriptionname

The header for the column showing the long form. This is defined as:

```
\newcommand{\glslongextraLongHeader}{%
 \descriptionname}
```

\glslongextraShortTargetFmt{⟨*entry-label*⟩}

This governs the way that the short form should be displayed, including the target. This is defined as:

```
\newcommand{\glslongextraShortTargetFmt}[1]{%
 \glsentryitem{#1}%
 \glstarget{#1}
  {{\glsxtrshort[noindex,hyper=false]{#1}}}%
 \glsxtrpostnamehook{#1}}
```

Note that the post-name hook is included.

\glslongextraLongFmt{⟨*entry-label*⟩}

This governs the way that the long form should be displayed. This is defined as:

```
\newcommand{\glslongextraLongFmt}[1]{%
 {\glsxtrlong
  [noindex,hyper=false]{#1}}\glspostdescription
}
```

Note that the post-description hook is included.

\glslongextraSubShortTargetFmt{⟨level⟩}{⟨entry-label⟩}

This governs the way that the short form for child entries should be displayed, including the target. This is defined as:

```
\newcommand{\glslongextraSubShortTargetFmt}[2]{%
 \glssubentryitem{#2}%
 \glstarget{#2}
  {{\glsxtrshort[noindex,hyper=false]{#2}}}%
 \glsxtrpostnamehook{#2}}
```

\glslongextraSubLongFmt{⟨level⟩}{⟨entry-label⟩}

This governs the way that the long form for child entries should be displayed. This is defined as:

```
\newcommand{\glslongextraSubLongFmt}[2]{%
 \glslongextraLongFmt{#2}}
```

\glslongextraShortNoNameSetDescWidth

This is used to compute the value of \glsdescwidth and assumes that none of the short forms are wider than \glslongextraShortHeader.

abbr−short−long

A two column style. The short form is in the left column. The long form is in the right. The location list isn't shown.

The table header is given by:

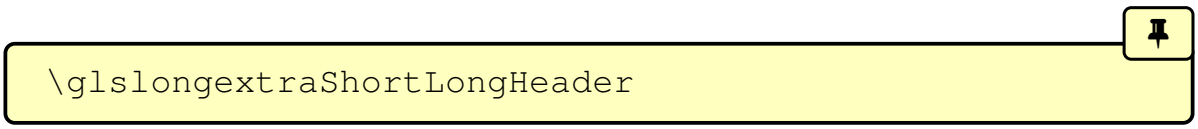
\glslongextraShortLongTabularHeader

which shows the column headers with horizontal rules.

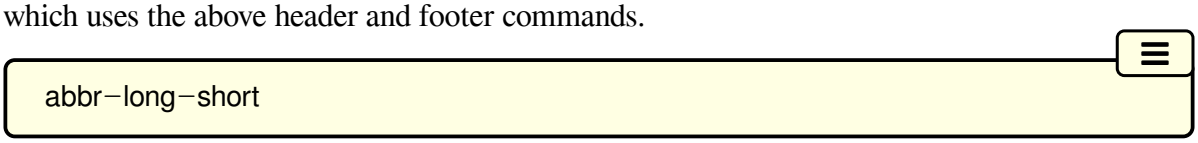
The table footer is given by:

\glslongextraShortLongTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

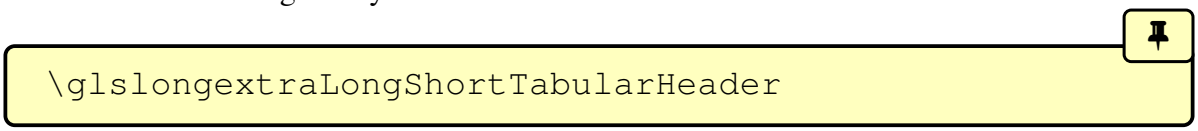


which uses the above header and footer commands.



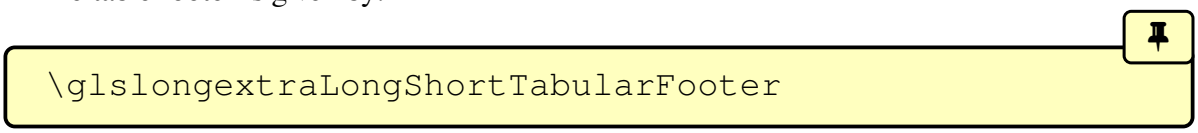
A two column style. The short form is in the right column. The long form is in the left. The location list isn't shown.

The table header is given by:

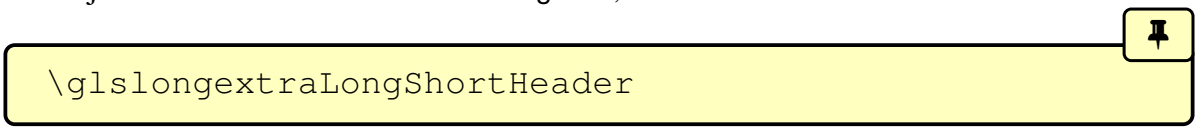


which shows the column headers with horizontal rules.

The table footer is given by:



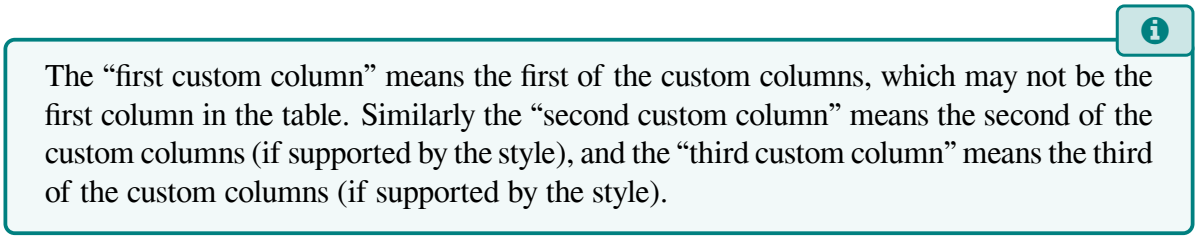
which just does a horizontal rule. With longtable, the table header and footer are set with:

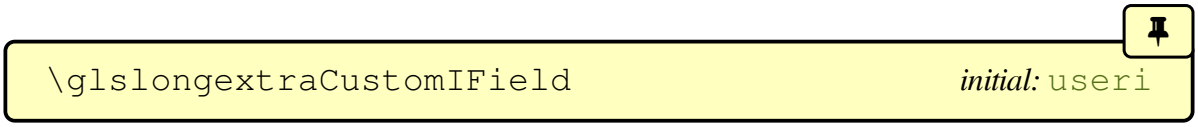


which uses the above header and footer commands.

8.7.2.7. Custom Fields

These styles allow one, two or three custom columns in addition to the name column. The "custom1" styles indicate one custom column, the "custom2" styles indicate two custom columns, and the "custom3" styles indicate three custom columns. Some styles also include the description column. These styles don't display the location. However, if you are using `bib2gls` you can set one of the custom fields to `location`, but if you have long location lists you may need to change the corresponding alignment command to switch to a paragraph column.





Expands to the internal field label for the first custom column. This will be used in the “custom1”, “custom2” and “custom3” styles.

```
\glslongextraCustomIIField                    initial: userii
```

Expands to the internal field label for the second custom column. This will be used in the “custom2” and “custom3” styles.

```
\glslongextraCustomIIIField                   initial: useriii
```

Expands to the internal field label for the third custom column. This will be used in the “custom3” style.

```
\glslongextraCustomIHeader
```

Expands to the header text for the first custom column. The default definition is:

```
\MFUsentencecase{\glslongextraCustomIField}
```

which means that it will be “Useri” by default, which is unlikely to be appropriate, but it may be suitable if you change the first custom field.

```
\glslongextraCustomIIHeader
```

Expands to the header text for the second custom column. The default likewise simply applies sentence case to the internal field label.

```
\glslongextraCustomIIIHeader
```

Expands to the header text for the third custom column. The default likewise simply applies sentence case to the internal field label.

```
\glslongextraCustomIFmt{⟨entry-label⟩}
```

This is used to format each top-level entry in the first custom column. The default definition is:

```
\glsxtrusefield{⟨entry-label⟩}{\glslongextraCustomIField}
```

```
\glslongextraSubCustomIFmt{⟨level⟩}{⟨entry-label⟩}
```

This is used to format each sub-entry in the first custom column. The default definition is:

```
\glslongextraCustomIIFmt{⟨entry-label⟩}
```

```
\glslongextraCustomIIIFmt{⟨entry-label⟩}
```

This is used to format each top-level entry in the second custom column. The default definition is:

```
\glsxtrusefield{⟨entry-label⟩}{\glslongextraCustomIIField}
```

```
\glslongextraSubCustomIIFmt{⟨level⟩}{⟨entry-label⟩}
```

This is used to format each sub-entry in the second custom column. The default definition is:

```
\glslongextraCustomIIFmt{⟨entry-label⟩}
```

```
\glslongextraCustomIIIFmt{⟨entry-label⟩}
```

This is used to format each top-level entry in the third custom column. The default definition is:

```
\glsxtrusefield{⟨entry-label⟩}{\glslongextraCustomIII-
Field}
```

```
\glslongextraSubCustomIIIFmt{⟨level⟩}{⟨entry-label⟩}
```

This is used to format each sub-entry in the third custom column. The default definition is:

```
\glslongextraCustomIIIFmt{⟨entry-label⟩}
```

```
\glslongextraCustomIAlign                    initial: l
```

Expands to the alignment specifier for the first custom column.

```
\glslongextraCustomIIAlign                   initial: l
```

Expands to the alignment specifier for the second custom column.

\glslongextraCustomIIIAlign *initial:* l

Expands to the alignment specifier for the third custom column.

\glslongextraCustomTabularFooter

The footer used for all the custom styles. The default definition simply does \bottomrule.

\glslongextraNameCustomITabularHeader

The table header for the long-name-custom1 style (which has two columns). This command and the previous command are used in the following.

\glslongextraNameCustomIHeader

The longtable header and footer markup for the long-name-custom1 style.

\glslongextraCustomINameTabularHeader

The table header for the long-custom1-name style (which has two columns). Used in the following.

\glslongextraCustomINameHeader

The longtable header and footer markup for the long-custom1-name style.

\glslongextraNameCustomIITabularHeader

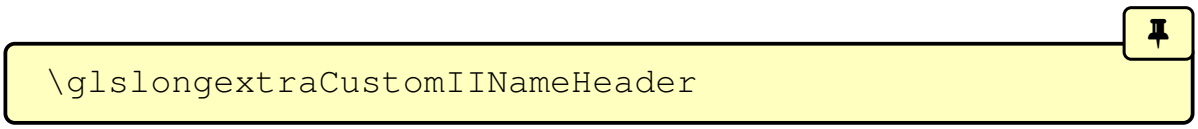
The table header for the long-name-custom2 style (which has three columns). Used in the following.

\glslongextraNameCustomIIHeader

The longtable header and footer markup for the long-name-custom2 style.

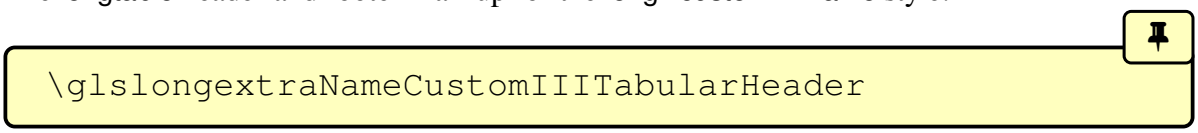
\glslongextraCustomIINameTabularHeader

The table header for the long-custom2-name style (which has three columns). Used in the following.



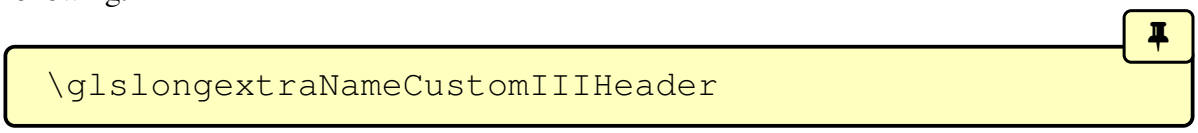
\glslongextraCustomIINameHeader

The longtable header and footer markup for the long−custom2−name style.



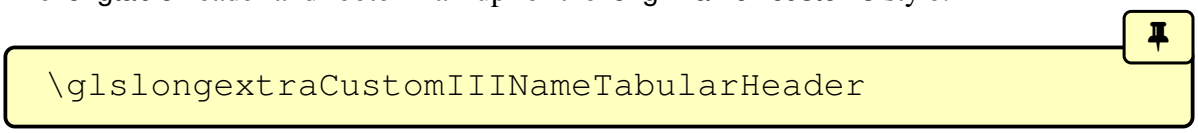
\glslongextraNameCustomIIITabularHeader

The table header for the long−name−custom3 style (which has four columns). Used in the following.



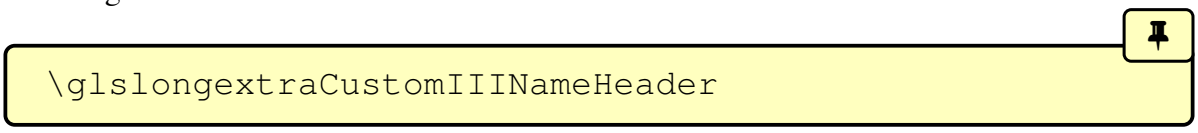
\glslongextraNameCustomIIIHeader

The longtable header and footer markup for the long−name−custom3 style.



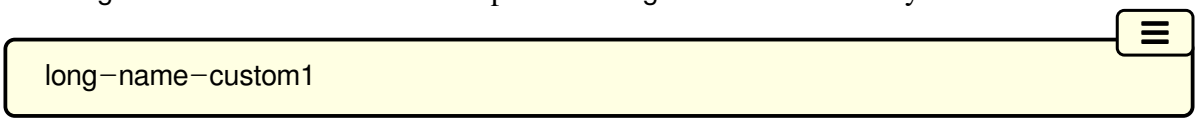
\glslongextraCustomIIINameTabularHeader

The table header for the long−custom3−name style (which has four columns). Used in the following.



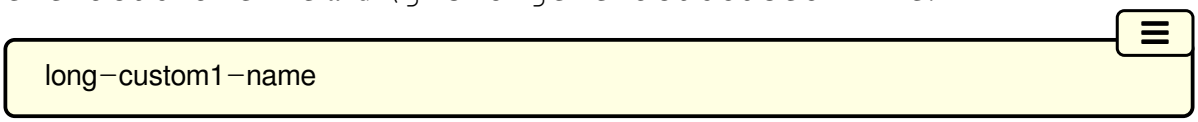
\glslongextraCustomIIINameHeader

The longtable header and footer markup for the long−custom3−name style.



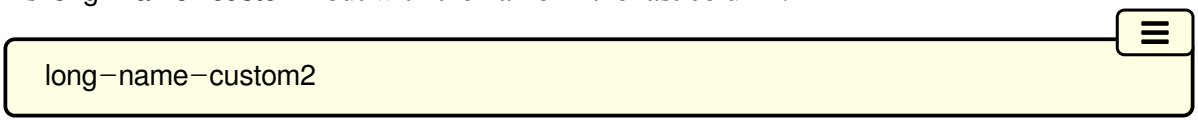
long−name−custom1

A two column style with the name in the first column and the first custom field in the second. For top-level entries, the name is formatted with \glslongextraNameFmt and the custom field is formatted with \glslongextraCustomIFmt. Sub-entries use \glslongextraSubNameFmt and \glslongextraSubCustomIFmt.



long−custom1−name

As long−name−custom1 but with the name in the last column.



long−name−custom2

A three column style with the name in the first column, the first custom field in the second and the second custom field in the third. As long−name−custom1, but additionally the second custom field is formatted with \glslongextraCustomIIFmt for top-level entries and with \glslongextraSubCustomIIFmt for child-entries.

long−custom2−name

As long−name−custom2 but with the name in the last column.

long−name−custom3

A four column style with the name in the first column, the first custom field in the second, the second custom field in the third, and the third custom field in the fourth. As long−name−custom2, but additionally the third custom field is formatted with `\glslongextraCustomIIIFmt` for top-level entries and with `\glslongextraSubCustomIIIFmt` for child-entries.

long−custom3−name

As long−name−custom3 but with the name in the last column.

The following styles also have a description column, which uses `\glslongextraDescAlign` for the column alignment. These styles attempt to calculate an appropriate width for `\glsdescwidth`.

`\glslongextraCustomISetDescWidth`

Sets `\glsdescwidth` for the long−name−custom1−desc style. This first uses `\glslongextraSetDescWidth` to calculate the width W if there were only a name and description column. It then measures the width of the first custom column header w and sets `\glsdescwidth` to $w - 2\backslash\texttt{tabcolsep} - w$. This assumes that the first custom column header is wider than the value of each entry's first custom field. If this isn't the case, then you will need to redefined this command as appropriate.

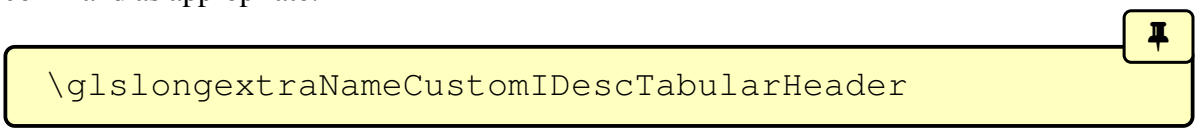
`\glslongextraCustomIISetDescWidth`

Sets `\glsdescwidth` for the long−name−custom2−desc style. This first uses `\glslongextraCustomISetDescWidth` to calculate the width W if there were only a name column, first custom column, and description column. It then measures the width of the second custom column header w and sets `\glsdescwidth` to $w - 2\backslash\texttt{tabcolsep} - w$. This assumes that the second custom column header is wider than the value of each entry's second custom field. If this isn't the case, then you will need to redefined this command as appropriate.

`\glslongextraCustomIIISetDescWidth`

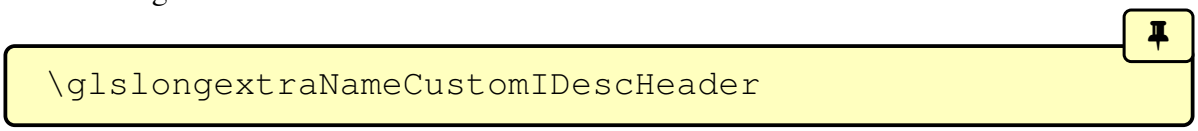
Sets `\glsdescwidth` for the long−name−custom3−desc style. This first uses `\glslongextraCustomIISetDescWidth` to calculate the width W if there were only a name column, first custom column, second custom column, and description column. It then

measures the width of the third custom column header w and sets \glsdescwidth to $w - 2\tabcolsep - w$. This assumes that the third custom column header is wider than the value of each entry's third custom field. If this isn't the case, then you will need to redefined this command as appropriate.



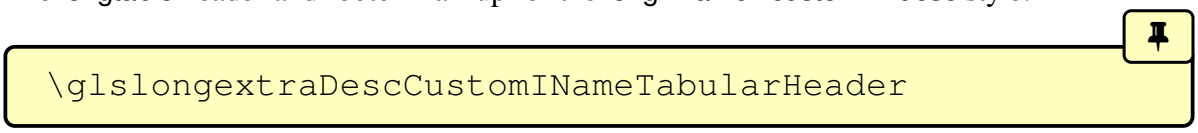
\glslongextraNameCustomIDescTabularHeader

The table header for the long-name-custom1-desc style (which has three columns). Used in the following.



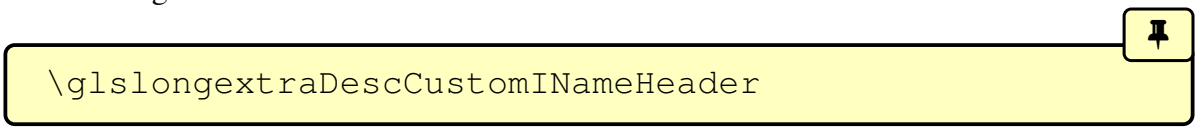
\glslongextraNameCustomIDescHeader

The longtable header and footer markup for the long-name-custom1-desc style.



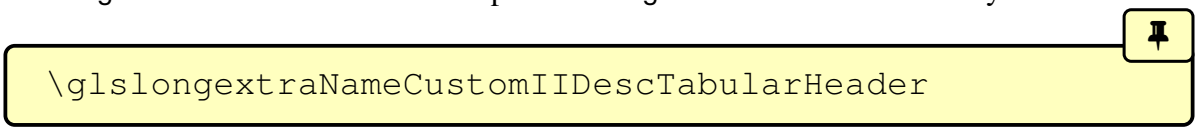
\glslongextraDescCustomINameTabularHeader

The table header for the long-desc-custom1-name style (which has three columns). Used in the following.



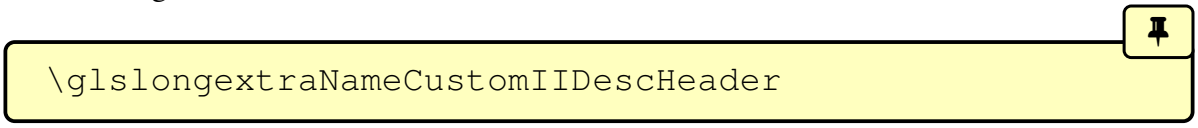
\glslongextraDescCustomINameHeader

The longtable header and footer markup for the long-desc-custom1-name style.



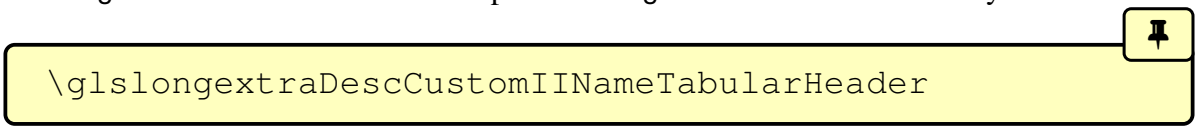
\glslongextraNameCustomIIDescTabularHeader

The table header for the long-name-custom2-desc style (which has four columns). Used in the following.



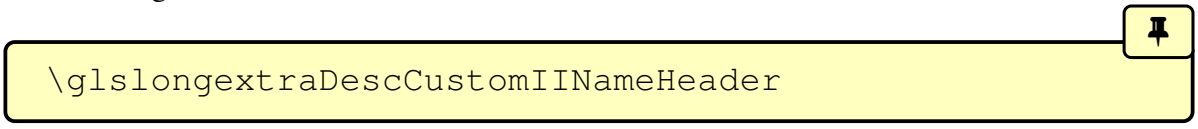
\glslongextraNameCustomIIDescHeader

The longtable header and footer markup for the long-name-custom2-desc style.



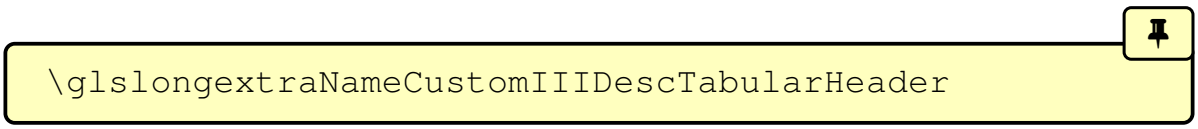
\glslongextraDescCustomIINameTabularHeader

The table header for the long-desc-custom2-name style (which has four columns). Used in the following.

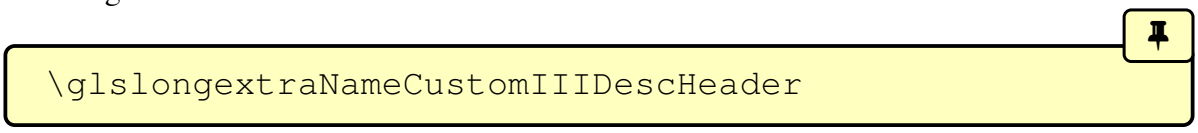


\glslongextraDescCustomIINameHeader

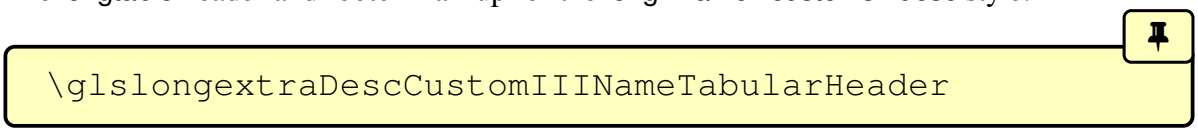
The longtable header and footer markup for the long-desc-custom2-name style.



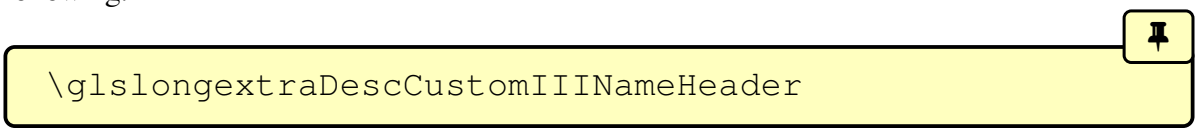
The table header for the long−name−custom3−desc style (which has five columns). Used in the following.



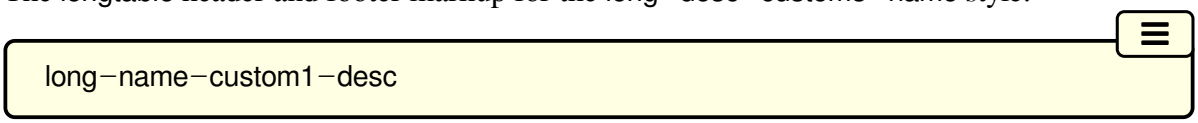
The longtable header and footer markup for the long−name−custom3−desc style.



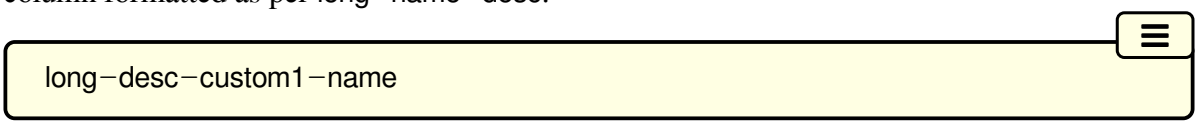
The table header for the long−desc−custom3−name style (which has five columns). Used in the following.



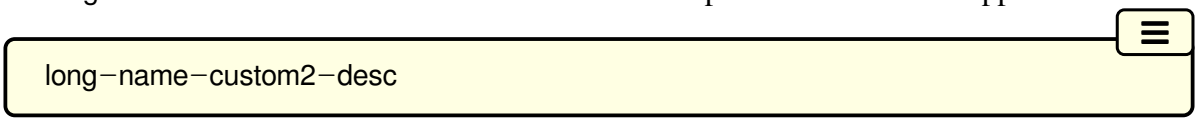
The longtable header and footer markup for the long−desc−custom3−name style.



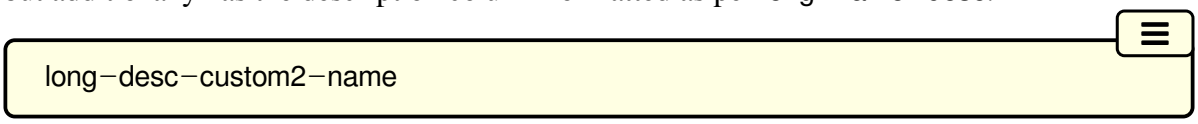
A three column style with the name in the first column, the first custom field in the second, and the description in the third. This is like long−name−custom1 but additionally has the description column formatted as per long−name−desc.



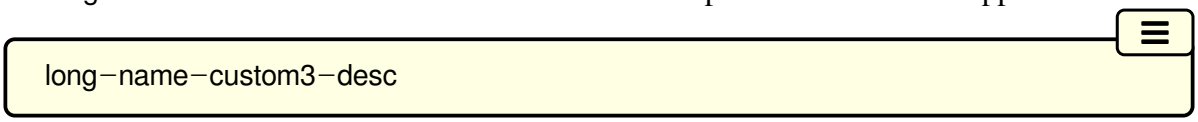
As long−name−custom1−desc but the name and description columns are swapped.



A four column style with the name in the first column, the first custom field in the second, the second custom field in the third, and the description in the fourth. This is like long−name−custom2 but additionally has the description column formatted as per long−name−desc.



As long−name−custom2−desc but the name and description columns are swapped.



A five column style with the name in the first column, the first custom field in the second, the

second custom field in the third, the third custom field in the fourth, and the description in the fifth. This is like long-name-custom3 but additionally has the description column formatted as per long-name-desc.

long-desc-custom3-name

As long-name-custom3-desc but the name and description columns are swapped.

8.7.3. glossary-topic package

The glossary-topic package provides glossary styles designed for hierarchical glossaries where the top-level entries are topic titles. This package automatically loads the multicol package. If the glossary-tree package is also loaded then commands like `\glssetwidest` can be used on these styles in much the same way as for the alttree style. If a widest value isn't set then these styles behave more like the tree style.

This package provides styles designed for glossaries that are lists of topics. That is, the top-level entries are considered topic titles (which may or may not have an associated symbol or description) and the sub-entries are items within that topic. By default the location list isn't shown for the top-level entries but is shown after the description for sub-entries (unless suppressed with `nonumberlist` or `save-locations=false`).

The following styles are provided:

topic

This style is similar to the tree style but the indentation doesn't start until the second sub-item level. The top-level entries have the name displayed in a larger font with the description following in a new paragraph (see Example 148). This style doesn't support the `nogroupskip` setting.

topicmcols

This style is like the topic style but the sub-entries are placed inside a multicols environment (unlike styles such as mcoltree where the entire glossary content is within a single multicols environment). The environment name is supplied in the value of the command:

```
\glstopicColsEnv
```

This defaults to `multicols`. You can change this to the starred form. For example:

```
\renewcommand{\glstopicColsEnv}{multicols*}
```

The number of columns is given by the command:

Example 148: The `topic` style

Glossary

Pictograph

Picture or symbol representing a word or phrase.

copy.
cut.
edit.
paste.

Symbols

Mathematical constants or functions.

constant a fixed quantity or numerical value.

π ratio of the length of the circumference of a circle to its diameter.

$\sqrt{2}$ Pythagoras' constant.

function a rule that assigns a value to every element of the domain.

$\cos(x)$ cosine.

$\ln(x)$ natural logarithm.

parameter a constant or variable that distinguishes a specific form.

x the abscissa value.

y the ordinate value.

z the applicate value.

```
\glstopicCols
```

This expands to 2, by default. This style is demonstrated in Example 149.

Example 149: The `topicmcols` style

Glossary

Pictograph

Picture or symbol representing a word or phrase.

copy.
cut.
edit.
paste.

Symbols

Mathematical constants or functions.

constant a fixed quantity or numerical value.

π ratio of the length of the circumference of a circle to its diameter.

$\sqrt{2}$ Pythagoras' constant.

function a rule that assigns a value to every element of the domain.

$\cos(x)$ cosine.

$\ln(x)$ natural logarithm.

parameter a constant or variable that distinguishes a specific form.

x the abscissa value.

y the ordinate value.

z the applicate value.

Both styles can have a widest name set like the alttree style, using the commands provided by glossary-tree and glossaries-extra-stylemods or with the `set-widest` resource option. If a widest name is set, then the sub-entry names will be placed in a box of the given width otherwise they won't be placed in a box. In Example 150, the widest names have been set for level 1 and level 2 using:

```
\glssetwidest[1]{parameter}
\glssetwidest[2]{\glsentryname{cosine}}
```

Note that this doesn't change the indentation at the start of the level 2 items to match the width of the level 1 widest name.

Both of the above styles use the following commands.

Example 150: The `topicmcols` style with the widest name set

Glossary

Pictograph

Picture or symbol representing a word or phrase.

	copy.		edit.
	cut.		paste.

Symbols

Mathematical constants or functions.

constant a fixed quantity or numerical value.

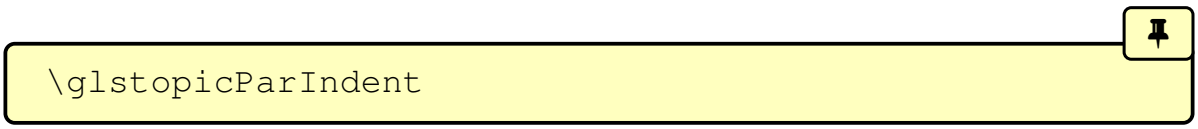
- π ratio of the length of the circumference of a circle to its diameter.
- $\sqrt{2}$ Pythagoras' constant.

function a rule that assigns a value to every element of the domain.

- $\cos(x)$ cosine.
- $\ln(x)$ natural logarithm.

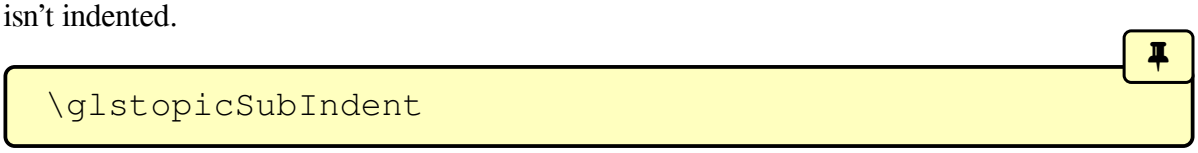
parameter a constant or variable that distinguishes a specific form.

- x the abscissa value.
- y the ordinate value.
- z the applicate value.



```
\glstopicParIndent
```

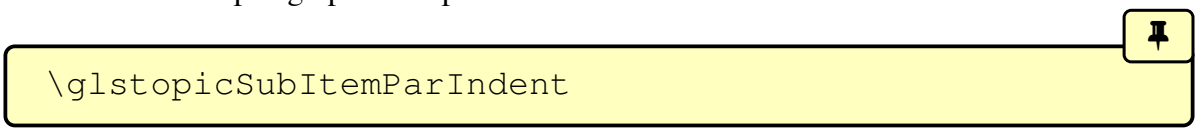
This command is a length that's used for the paragraph indentation in any multi-paragraph description for top-level entries, but not for the first paragraph (at the start of the description) which isn't indented.



```
\glstopicSubIndent
```

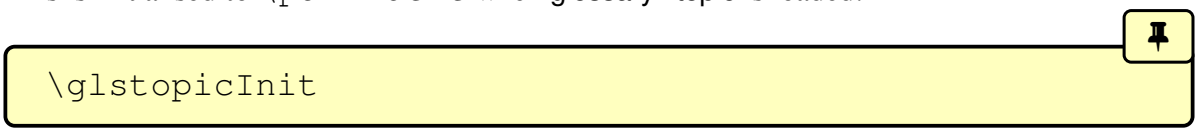
This command is a length that's used to calculate the hanging indentation for sub-entries. The level 1 sub-entries don't indent the name. Level n sub-entries have the name indented by $(n-1) \times$ `\glstopicSubIndent`. The hanging indent depends on whether or not a widest name has been set for the level.

There is also a length for additional indentation used in the second paragraph onwards for child entries with multi-paragraph descriptions:



```
\glstopicSubItemParIndent
```

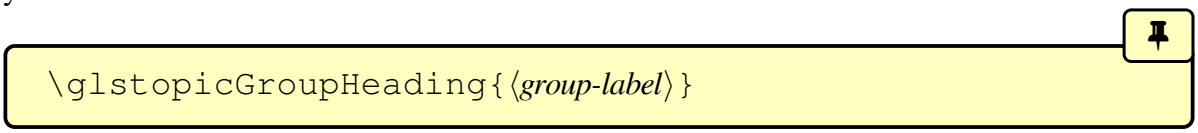
This is initialised to `\parindent` when glossary-topic is loaded.



```
\glstopicInit
```

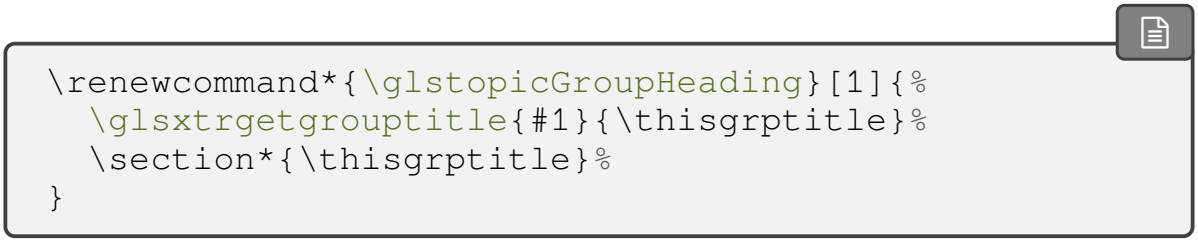
This hook is used at the start of the glossary. It does nothing by default.

Although the styles don't support letter groups by default, if you have many topics (top-level entries) and you feel that it would help the reader to divide them up into headed letter groups, you can redefine:



```
\glstopicGroupHeading{⟨group-label⟩}
```

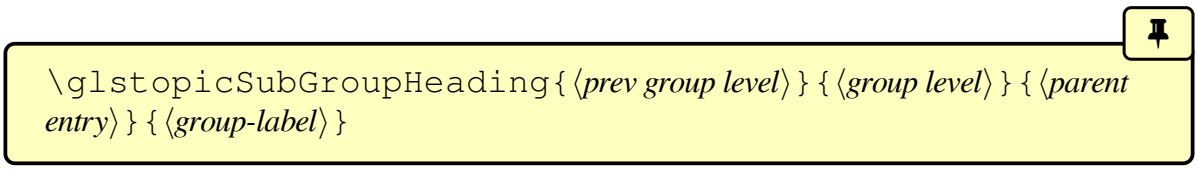
This does nothing by default. If you want to redefine it, you can fetch the title corresponding to the group label with `\glsxtrgetgrouptitle`. For example:



```
\renewcommand*{\glstopicGroupHeading}[1]{%
  \glsxtrgetgrouptitle{#1}{\thisgrptitle}%
  \section*{\thisgrptitle}%
}
```

Remember that if you are using `bib2gls`, you will need the `--group` or `-g` switch to support groups (see §8.4.1).

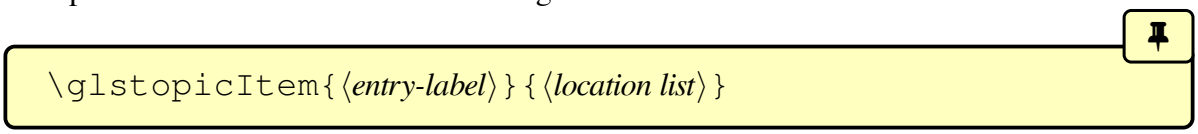
Sub-groups are only available with `bib2gls` and the `group-level` option. If they are supported, sub-group headings are formatted according to:



\glstopicSubGroupHeading{⟨*prev group level*⟩}{⟨*group level*⟩}{⟨*parent entry*⟩}{⟨*group-label*⟩}

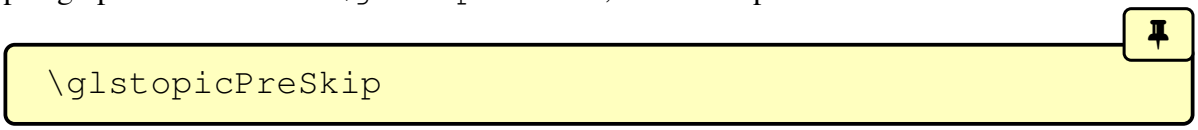
This formats the sub-group heading. Note that unlike \glstopicGroupHeading this command does actually format the sub-group heading by default. This means that it you use group-level=all, the top-level groups won't be displayed, but the sub-groups will be.

Top-level entries are formatted according to:



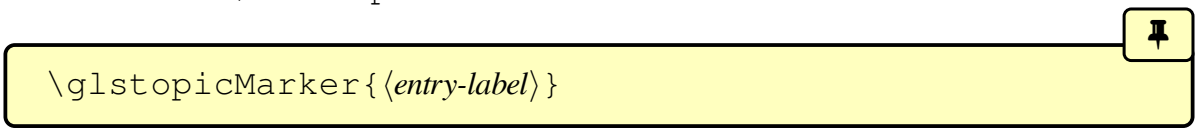
\glstopicItem{⟨*entry-label*⟩}{⟨*location list*⟩}

This formats the name and (if provided) the symbol. The description (if set) will follow in a new paragraph. At the start of \glstopicItem, a vertical space is added with:



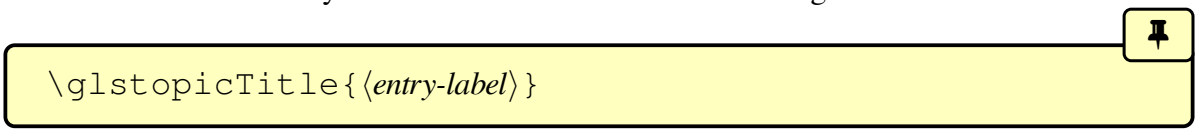
\glstopicPreSkip

which defaults to \medskip. There is then a hook:



\glstopicMarker{⟨*entry-label*⟩}

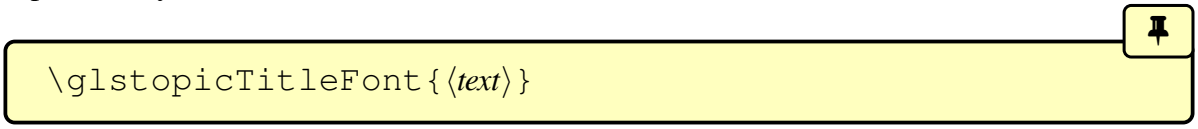
which does nothing by default, but may be redefined. For example, to add a line to the table of contents. The name and symbol are set in the form of a title using:



\glstopicTitle{⟨*entry-label*⟩}

This uses \Glossentryname to apply sentence case. (Note, if your descriptions have paragraph breaks make sure that you have mfirstuc v2.09+.)

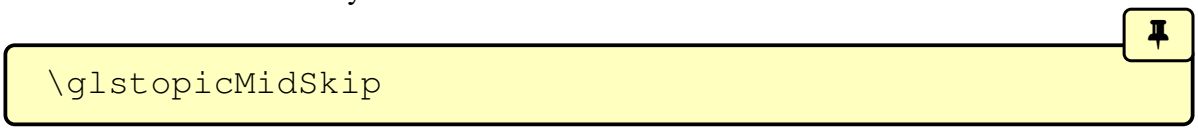
If there's a symbol, this is added in parentheses. Both name and symbol (if present) are encapsulated by:



\glstopicTitleFont{⟨*text*⟩}

This uses a bold, large font by default.

If the entry has the description key set (tested with \ifglshasdesc) then a paragraph break is inserted followed by:



\glstopicMidSkip

which defaults to \smallskip. This is followed by the description which is formatted according to:

```
\glstopicDesc{⟨entry-label⟩}
```

This just does `\Glossentrydesc{entry-label}` followed by the post-description hook. There is then a paragraph break followed by:

```
\glstopicPostSkip
```

regardless of whether or not the description was displayed. This defaults to `\smallskip`. This is then followed by:

```
\glstopicLoc{⟨entry-label⟩}{⟨location list⟩}
```

which may be used to display the location list, but does nothing by default.

The sub-entries first set up the paragraph and hanging indentations using:

```
\glstopicAssignSubIndent{⟨level⟩}
```

This uses:

```
\glstopicAssignWidest{⟨level⟩}
```

to determine if a widest name has been set for the given level.

The sub-entry has its information displayed using:

```
\glstopicSubItem{⟨level⟩}{⟨entry-label⟩}{⟨location list⟩}
```

This encapsulates the name with:

```
\glstopicSubNameFont{⟨text⟩}
```

By default this just uses `\textbf`. This is followed by:

```
\glstopicSubItemSep
```

which defaults to `\quad`. The name and separator are passed in the ⟨*text*⟩ argument of:

```
\glstopicSubItemBox{⟨level⟩}{⟨text⟩}
```

If a widest name was set for the given level, this will put ⟨*text*⟩ inside a box of that width otherwise it just does ⟨*text*⟩.

This is followed by the symbol in parentheses if set. Then, if the description is set, the description and post-description hook are displayed followed by:

```
\glstopicSubPreLocSep
```

(This command isn't used if the description isn't set.)

Finally the location list is displayed using:

```
\glstopicSubLoc{⟨entry-label⟩}{⟨location list⟩}
```

which just does ⟨*location*⟩ by default.

8.7.4. glossary−table package

The glossary−table package is new to version 1.49. It automatically loads the longtable, array and booktabs packages. If you want to use `\glspenaltygroupskip` for the group skip, you will need to also load glossary−longbooktabs.

> This package is designed specifically for use with `bib2gls`. It can be used to create a supplemental glossary with other indexing options, but the entries will be listed in order of definition and no child entries will be shown.

The glossary−table package doesn't provide any general purpose styles, but instead provides one highly customized style (table), which is designed to work with a supplied command:

```
\printunsrttable[⟨options⟩]
```

The table style should not be set with the `style` package option, `\setglossarystyle` or the `style` option, as it's only intended for use within `\printunsrttable`, which sets up the appropriate hooks to allow the style to work with `\printunsrtglossary` (which is used implicitly).

Tabular styles such as `long` create a longtable with one entry per row and no caption. The longheader style is similar but adds a header row, and the long−booktabs style includes rules above and below the header row and at the end of the table. In all these longtable styles, the glossary title is outside of the style, and is typically put in a sectioning command. Similarly, the glossary preamble `\glossarypreamble` and postamble `\glossarypostamble` are outside of longtable.

The table style, on the other hand, allows multiple entries per row. The glossary title (`title`) is the table caption with what's normally the table of contents title (`toctitle`) as the caption

title for the list of tables. Similarly, the preamble and postamble are included in the table header and footer, instead of being outside of the table.

This means that `\glossarysection`, `\glossarypreamble` and `\glossarypostamble` are redefined by `\printunsrttable` to do nothing so that they aren't shown outside of the longtable by `\printunsrtglossary`, otherwise there would be a duplication of the information in the header and footer of the table. The `\printunsrtglossary` hooks are used to insert the inter-block tabulation (`&`) character and new row command in the construction performed outside of longtable, which would otherwise cause issues if used directly in the table style.

The block styles (see §8.7.4.3) alter the way the table style sets up the longtable environment and the way that the entries are formatted. The top level glossary style command `\glossentry` is defined to do the block according to the designated block style, which includes the child entries, if the `childcount` field has been set and is non-zero.

The `\subglossentry` command is redefined to do nothing, but it won't be used as the child entries are all filtered out. If you don't use the `save-child-count` resource option, no child entries will be shown. There's no recursive descent down the hierarchical levels.

This means that the child entries will be listed in one of the columns in the block, according to the style. This can make the column quite wide. The child names aren't displayed by default but the block styles support the `subentrycounter` option. The child entries are listed in a tabular environment, which means they are contained in the same row as their parent and can't be broken across a page.

A "block" indicates a block of columns used to format one entry (and, optionally, its children). One row of the table may contain multiple blocks. For example, a block may consist of two columns with the name in the first column and the description in the second, or may consist of three columns with the name in the first column, the symbol in the second, and the description in the third. So if a block style has 3 columns, and the desired number of blocks is set to 2, then the table will have a total of $3 \times 2 = 6$ columns.

The style supports up to 1 hierarchical level, but you will need the `save-child-count` resource option if you want the level 1 sub-entries to show. Deeper level entries are omitted. Sub-entries are automatically filtered by a custom hook that `\printunsrtglossaryentryprocesshook` is assigned to within `\printunsrttable`. This custom hook allows additional filtering to be employed with the command:

`\glstableiffilter{`⟨*entry-label*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This command should do ⟨*true*⟩ if the entry identified by ⟨*entry-label*⟩ should be skipped, otherwise it should do ⟨*false*⟩. The default definition simply does ⟨*false*⟩.

For example, the following will filter entries that have the category set to `general`:

```
\renewcommand{\glstableiffilter}[1]{%
 \glsifcategory{#1}{general}}
```

Note that if this command is redefined to do neither ⟨*true*⟩ nor ⟨*false*⟩ or does both, it will interfere with the width calculations if `par` isn't set to the default `par=false`.

You can use the `init` option to locally redefine commands within `\printunsrttable`. For example:

```
\printunsrttable[init={%
 \renewcommand{\glstableiffilter}[1]%
  \glsifcategory{##1}{general}%
}]
```

An extra field (the "other" field) may be added with the `other` key. If this value is empty, then no extra field will be added. Some block styles, such as `other-name` and `symbol-other` put the other field in its own column. If the other field isn't set, this will lead to an empty column.

If there isn't a designated column for the other field, then block styles that show the description will put the other field in before the description, but in the same column as the description. Otherwise, block styles that don't show the description, will put the other field after the name, but in the same column as the name.

Example 151 uses the `name` block style, which only has one column per block. The name is followed by the description in parentheses (if set), which is then followed by the child list. I redefined `\glstableNameFmt` to make the name appear in bold, to highlight it. I've used the `par=ragged` option, otherwise the table will be too wide to fit the page. 151

```
\usepackage
 [record,stylemods=table,subentrycounter]
 {glossaries-extra}

\GlsXtrLoadResources[
 % entries in example-glossaries-childnoname.bib:
 src={example-glossaries-childnoname},
 selection=all,
 save-child-count]
\begin{document}
\printunsrttable
 [
  block-style=name,par=ragged,
```

```
  preamble={Some preamble text},
  postamble={Some postamble text},
  init={%
    \let\glstableNameFmt\textbf
    \def\glstablenameheader{Summary}%
  }
 ]
\end{document}
```

This creates a table with two entries per row.

Example 151: Two entries per row with `\printunsrttable`

Table 1: Glossary

Some preamble text

Summary	Summary
class (aptent taciti) 1) ad litora 2) sociosqu 3) torquent per conubia	**consectetuer** (mauris) 1) leo justo 2) quis egestas
cum sociis (natoque penatibus) 1) dis parturient montes 2) et magnis	**eleifend** (sit amet faucibus) 1) elementum 2) urna sapien
mauris (libero eros) 1) dapibus porttitor, pede 2) lacinia non 3) sodales quis	**non risus** (morbi non felis) 1) ac libero 2) vulputate fringilla
nostra (per inceptos hymenaeos) 1) mauris condimentum nulla 2) morbi dapibus	**scelerisque** (at) 1) sem leo 2) tellus et tortor 3) nec, enim 4) sit amet 5) vehicula pellentesque 6) facilisis id 7) eu, nulla

Note that each row is as deep as the entry with the most children. So where a row has one column with two children and another with seven, the row is deep enough to accommodate the seven child entries, which leaves a gap below the smaller list of two children.

8.7.4.1. Child Entries

Entries with a hierarchical level greater than 0 are filtered out (see above). This takes the `leveloffset` option into account. Child entries can be included, but only by checking if the `childcount` field has been set and is non-zero. This is done by:

```
\glstableChildEntries{⟨entry-label⟩}
```

Note that `\glstableiffilter` filters top-level entries, and their child entries will also be filtered. Child entries for non-filtered top-level entries can be filtered by redefining:

```
\glstableiffilterchild{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
```

where ⟨*entry-label*⟩ is the child entry label. This command should do ⟨*true*⟩ if the child entry should be filtered and ⟨*false*⟩ otherwise.

If the child count is non-zero, taking both `childcount` and child filtering into account, then `\glstableChildEntries` command will display the non-filtered children in the form:

```
\glstablePreChildren
\begin{glstablesubentries}
\glstableblocksubentry{⟨child-1-label⟩}
\glstableblocksubentrysep
\glstableblocksubentry{⟨child-2-label⟩}
...
\glstableblocksubentrysep
\glstableblocksubentry{⟨child-n-label⟩}
\end{glstablesubentries}
```

This consists of the following.

```
\glstablePreChildren
```

Occurs at the start. If `par=justified` or `par=ragged`, this will do `\par` otherwise it does nothing.

In general, it's best not to list children with `par=false`, except with a style like `name` or `name-desc` with no description, as the table can end up too wide for the page.

```
\begin{glstablesubentries}
⟨content⟩
\end{glstablesubentries}
```

This environment encapsulates the child list. By default, this does:

```
\begin{tabular}[t]{⟨align⟩}
⟨content⟩
\end{tabular}
```

The ⟨*align*⟩ argument is obtained by expanding:

```
\glstablesubentryalign
```

which takes the `par` setting into account.

Each child item is display using `\glstableblocksubentry` which is redefined by the block style.

The separator between each child item is given by:

```
\glstableblocksubentrysep
```

This just expands to `\glstablenewline`.

8.7.4.2. Options

The optional argument of `\printunsrttable` may have the options that can typically be passed to `\printunsrtglossary`, except that the `nonumberlist` and `style` options won't have an effect. If you want the location list, it can simply be obtained from the `location` field in the appropriate style hook.

Some default settings are changed: `groups=false` and `nogroupskip`true. If you want letter group headings, you will need to both add `groups=true` to the options list and invoke `bib2gls` with the `--group` switch. The group headings will span the entire width of the table. This may result in empty blocks at the end of the previous row. If you want a vertical gap before the group heading (but not before the first group), you will need to add `nogroupskip`false, but you will also need to load glossary−longbooktabs. Note that this option is designed to be used with group headings and will have no effect with `groups=false`.

Additionally, the following options may also be used.

`blocks`=⟨*n*⟩ *initial:* **2**

The value is the number of blocks in the table. The total number of columns in the table will be this value multiplied by the number of columns per block, which is determined by the block

style. For example, the `name-desc` block style has two columns, so if there are three blocks then there will be a total of six columns.

`caption`=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`true`**

A boolean option that determines whether or not to include a caption. The caption on the first page of the table is produced with:

```
\glstablecaption{⟨lot title⟩}{⟨title⟩}{⟨label code⟩}
```

where ⟨*label code*⟩ is the code to create the label, if one has been supplied (either by an option such as `numberedsection=autolabel` or by the `label` option). The ⟨*title*⟩ argument will be `\glossarytitle`, which can be changed with the `title` option. The ⟨*lot title*⟩ argument is the title for the list of tables and is actually what would normally be the title for the table of contents, which can be set with the `toctitle` option. The default definition simply does:

```
\caption[⟨lot title⟩]{⟨label code⟩⟨title⟩}
```

An empty ⟨*lot title*⟩ (`toctitle={}`) will prevent the caption from being added to the list of tables.

The `numberedsection` option will only influence the label, not the table numbering. If you don't want the table numbered, redefine `\glstablecaption` to use `\caption*`.

If the table spans across multiple pages, the caption for subsequent pages will be produced with:

```
\glstablenextcaption{⟨lot title⟩}{⟨title⟩}
```

This ignores the ⟨*lot title*⟩ argument by default and does:

```
\caption[]{⟨title⟩\glstablepostnextcaption}
```

This has an empty optional argument to prevent the caption from being repeatedly added to the list of tables. The title is followed by:

`\glstablepostnextcaption` *initial:* `␣Cont./`

You can either redefine this command to adjust the content after the title or redefine `\glstablenextcaption`, as appropriate.

header=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

A boolean option to determine whether or not to show the header row. Note that a header with three column block styles, such as `name-symbol-desc`, can result in overfull lines. You may need to shorten the header text to fit.

The header text is produced with one of the following commands:

```
\glstablenameheader
```

Expands to the header for the name column. Just uses `\entryname` by default.

```
\glstabledescheader
```

Expands to the header for the description column for block styles like `name-desc` and `name-symbol-desc`. Just uses `\descriptionname` by default.

```
\glstablesymbolheader
```

Expands to the header for the symbol column for block styles like `name-symbol` and `name-symbol-desc`. Just uses `\symbolname` by default.

```
\glstableotherheader
```

Expands to the header for the other column. The default definition applies `\MFUsentencecase` to the other field label.

```
\MFUsentencecase{\glstableotherfield}
```

rules=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

A boolean option to determine whether or not to show the horizontal rules (provided by booktabs). If used with `header=true`, there will be a rule above and below the header row. If used with `header=false`, there will only be one rule at the top of the table. In both cases, there will be a rule at the bottom of the table.

blocksep=⟨*alignment spec*⟩ *initial:* |

The value is inserted into the alignment specifier list between blocks. For example, the default value of the pipe character will insert a vertical line. Set this value to empty to remove it.

par=⟨*value*⟩ *initial:* **false**

Indicates whether or not the columns should be paragraphs. The value may be one of: `false`, `justified` or `ragged`. The default `par=false` will just use one of the column specifiers `l`, `r` or `c`. The other values will use the `p` specifier, in which case the column widths will be calculated.

other=⟨*field-label*⟩ *initial:* ***empty***

This should be set to the internal field label of the other field or to empty if no other field should be included.

init={⟨*code*⟩} *initial:* ***empty***

The ⟨*code*⟩ will be added shortly before `\printunsrtglossary` is called and any local changes will be scoped.

block-style=⟨*value*⟩ *initial:* **name-desc**

The block style. Available styles are listed in §8.7.4.3.

8.7.4.3. Block Styles

The block style may be set with the `block-style` option or with:

`\glstablesetstyle{`⟨*style-name*⟩`}`

The block styles are still under development, so the underlying commands are not yet documented and liable to change.

The following block styles are predefined.

block-style=**name**

Blocks have one column with the name, which is followed by the symbol and the description, if they have been set, in parentheses. The child list follows at the end of the column (if `child-count` is set and non-zero).

block-style=name-desc

This is the default style. Blocks have two columns with the name in the first column of the block and the description in the second. If the other field is set, it will follow the description. The child entries will be at the end of the second column (if `childcount` is set and non-zero).

block-style=desc-name

As `name-desc` but with the columns swapped. The child entries (if `childcount` is set and non-zero) will be at the end of the first column.

block-style=name-symbol

Blocks have two columns with the name in the first column of the block and the symbol in the second. If the other field is set, it will be placed after the name in the first column. The child entries are at the end of the first column (if `childcount` is set and non-zero).

block-style=symbol-name

As `name-symbol` but with the columns swapped. The child entries (if `childcount` is set and non-zero) will be at the end of the second column.

block-style=name-other

This is like `name-desc` but puts the other field in the second column. The description and symbol aren't shown. The child entries (if `childcount` is set and non-zero) will be at the end of the second column.

block-style=other-name

This is like `desc-name` but puts the other field in the second column. The description and symbol aren't shown. The child entries (if `childcount` is set and non-zero) will be at the end of the first column.

block-style=symbol-other

This is like `name-other` but shows the symbol instead of the name. The child entries (if `childcount` is set and non-zero) will be at the end of the second column.

block-style=other-symbol

This is like `other-name` but shows the symbol instead of the name. The child entries (if `childcount` is set and non-zero) will be at the end of the first column.

block-style=name-symbol-desc

Blocks have three columns with the name in the first column of the block, the symbol in the second, and the description in the third, preceded by the other field, if set. The child entries are at the end of the third column (if `childcount` is set and non-zero).

block-style=name-desc-symbol

Blocks have three columns with the name in the first column of the block, the description in the second, preceded by the other field, if set, and the symbol in the third. The child entries are at the end of the second column (if `childcount` is set and non-zero).

block-style=name-other-desc

Blocks have three columns with the name in the first column of the block, the other in the second, and the description in the third. The child entries are at the end of the third column (if `childcount` is set and non-zero).

block-style=desc-other-name

Blocks have three columns with the description in the first column of the block, the other in the second, and the name in the third. The child entries are at the end of the first column (if `childcount` is set and non-zero).

block-style=name-symbol-other-desc

Blocks have four columns with the name in the first column of the block, the symbol in the second, the other in the third, and the description in the fourth. The child entries are at the end of the fourth column (if `childcount` is set and non-zero).

block-style=name-other-symbol-desc

Blocks have four columns with the name in the first column of the block, the other in the second, the symbol in the third, and the description in the fourth. The child entries are at the end of the fourth column (if `childcount` is set and non-zero).

block-style=desc-symbol-other-name

Blocks have four columns with the description in the first column of the block, the symbol in the second, the other in the third, and the name in the fourth. The child entries are at the end of the first column (if childcount is set and non-zero).

block-style=desc-other-symbol-name

Blocks have four columns with the description in the first column of the block, the other in the second, the symbol in the third, and the name in the fourth. The child entries are at the end of the first column (if childcount is set and non-zero).

8.7.4.4. Associated Commands

The rows are separated with:

```
\glstablenewline
```

This simply does \tabularnewline (not \\ which has a different action in paragraph columns).

The following commands are used in the column specifier where a left, right or centred column is required, taking the par option into account. Note that with par=justified, the result will always be p{⟨*width*⟩}, whereas with par=ragged the paragraph will be ragged right or ragged left or have centring applied.

```
\glstableleftalign{⟨width⟩}
```

Expands to l or p{⟨*width*⟩} or >\protect\raggedrightp{⟨*width*⟩}, depending on the par setting.

This command is used in the column specifier where a left-justified column is required.

```
\glstablerightalign{⟨width⟩}
```

Expands to r or p{⟨*width*⟩} or >\protect\raggedleftp{⟨*width*⟩}, depending on the par setting.

This command is used in the column specifier where a right-justified column is required.

```
\glstablecenteralign{⟨width⟩}
```

Expands to c or p{⟨*width*⟩} or >\protect\centeringp{⟨*width*⟩}, depending on the par setting.

This command is used in the column specifier where a centred column is required.

```
\glstablenamecolalign
```

Expands to the alignment for the name column. The default definition uses left alignment:

```
\glstableleftalign{\glstablenamewidth}
```

```
\glstabledesccolalign
```

Expands to the alignment for the description column. The default definition uses left alignment:

```
\glstableleftalign{\glstabledescwidth}
```

```
\glstablesymbolcolalign
```

Expands to the alignment for the symbol column, in block styles where the symbol has its own column. The default definition uses centred alignment:

```
\glstablecenteralign{\glstablesymbolwidth}
```

```
\glstableothercolalign
```

Expands to the alignment for the other column, in block styles where the other field has its own column. The default definition uses left alignment:

```
\glstableleftalign{\glstableotherwidth}
```

If `par=justified` or `par=ragged`, the column widths will be calculated. The following length registers will be set, where applicable to the block style.

```
\glstablenamewidth
```

The width of the name column.

```
\glstabledescwidth
```

The width of the description column.

\glstablesymbolwidth

The width of the symbol column.

\glstableotherwidth

The width of the other column.

Unless par=false, the table will be the width of a line and each block will have equal width.

\glstableblockwidth

Note that in all the above, the width doesn't include the inter-column space given by \tabcolsep. The length registers below are initialise to 5pt, and can be redefined as appropriate.

\glstablepostpreambleskip *initial:* 5pt

The vertical skip after the preamble.

\glstableprepostambleskip *initial:* 5pt

The vertical skip before the postamble.

Formatting for the name, symbol, description and other field values are applied by the following commands.

\glstableNameFmt{⟨*text*⟩}

Formatting applied to the name. Simply does ⟨*text*⟩ by default. Note that the argument ⟨*text*⟩ will \glossentryname{⟨*label*⟩}, so any formatting applied by that command will also be in effect.

\glstableSubNameFmt{⟨*text*⟩}

Formatting applied to the child name. Does nothing by default, which means that the child name won't show.

\glstableSymbolFmt{⟨*text*⟩}

Formatting applied to the symbol. Simply does ⟨*text*⟩ by default. Note that the argument ⟨*text*⟩ will `\glossentrysymbol{`⟨*label*⟩`}`, so any formatting applied by that command will also be in effect.

`\glstableSubSymbolFmt{`⟨*text*⟩`}`

Formatting applied to the child symbol. Just does `\glstableSymbolFmt` by default.

`\glstableDescFmt{`⟨*text*⟩`}`

Formatting applied to the description. Simply does ⟨*text*⟩ by default. Note that the argument ⟨*text*⟩ will be:

```
\glossentrydesc{⟨label⟩}\glspostdescription
```

so any formatting applied by `\glossentrydesc` will also be in effect. Note that the post-description hook is included in the formatted.

`\glstableSubDescFmt{`⟨*text*⟩`}`

Formatting applied to the child description. Just does `\glstableDescFmt` by default.

The other field's internal label is provided by expanding:

`\glstableotherfield` *initial: empty*

This is redefined by the `other` option, but it may be redefined before `\printunsrtable` if a default field is required.

`\glstableOtherFmt{`⟨*text*⟩`}`

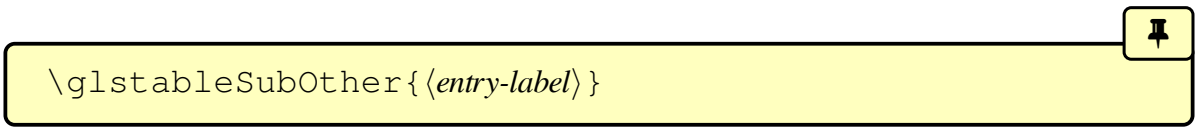
The formatting applied to the other field. This just does ⟨*text*⟩ by default. The field value itself is displayed with:

`\glstableOther{`⟨*entry-label*⟩`}`

The default definition does:

```
\glstableOtherFmt{%
 \glsxtrusefield{⟨entry-label⟩}{\glstableotherfield}}
```

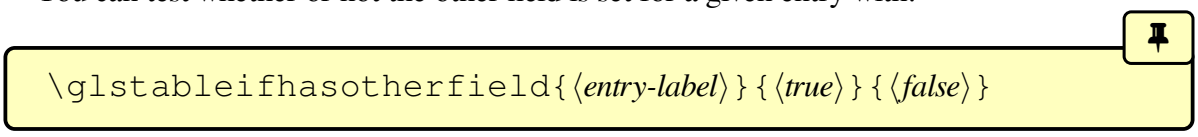
The value for the child entries is displayed with:



\glstableSubOther{⟨*entry-label*⟩}

The default definition simply does \glstableOther{⟨*entry-label*⟩}

You can test whether or not the other field is set for a given entry with:

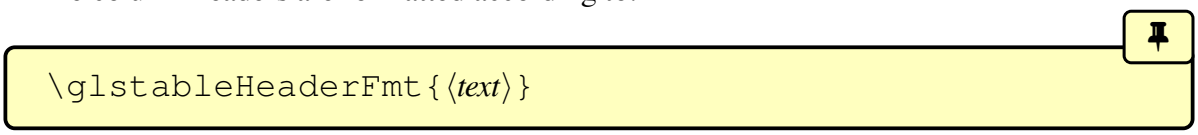


\glstableifhasotherfield{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}

This does ⟨*true*⟩ of the other field is non-void (according to \ifglsfieldvoid) otherwise it does ⟨*false*⟩. This will always do ⟨*false*⟩ if \glstableotherfield is void.

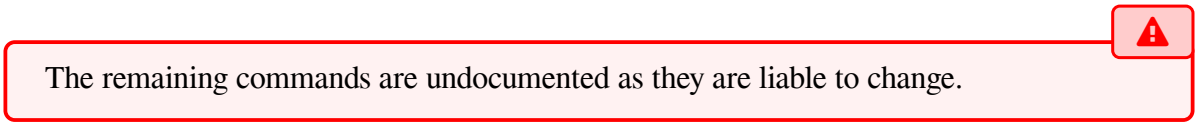
The column headers are supplied, where applicable, by the commands \glstablenameheader, \glstabledescheader, \glstablesymbolheader, and \glstableotherheader.

The column headers are formatted according to:



\glstableHeaderFmt{⟨*text*⟩}

The default definition is \textbf{⟨*text*⟩}.



The remaining commands are undocumented as they are liable to change.

9. Accessibility Support

The glossaries package comes with a supplementary package glossaries-accsupp that helps provide accessibility support. The glossaries-extra package provides additional support, but only if the glossaries-accsupp package has already been loaded when the relevant commands are defined. The best and simplest way to do this is through the `accsupp` package option.

See the "Accessibility Support" chapter in the glossaries user guide for further information about glossaries-accsupp.

9.1. Abbreviations

The accessibility fields relating to abbreviations are `shortaccess`, `shortpluralaccess`, `longaccess` and `longpluralaccess`. These provide the replacement text for the corresponding `short`, `shortplural`, `long` and `longplural` fields. The `access` field provides the replacement text for the `name` field.

Some of these accessibility fields are automatically assigned by `\newabbreviation` if they haven't been set.

```
\glsxtrassignactualsetup
```

This command is used to locally redefine common formatting commands so that they can be stripped to obtain only the text. You can add additional commands with `\appto`. For example, the following eccentric example has some strange styling in the abbreviation:

```
\newabbreviation{foo}
 {f\textsuperscript{o}\textsubscript{o}}
 {furry old otters}
```

If an accessibility field is being automatically assigned with text obtained from the short value, then the subscript and superscript commands will need to be stripped. These need to be locally redefined to just do their arguments:

```
\appto\glsxtrassignactualsetup{%
  \letcs{\textsuperscript}{@firstofone}%
  \letcs{\textsubscript}{@firstofone}%
```

```
}
```

The attributes that specifically relate to accessibility in abbreviations are listed below. The "actual short value" means the value obtained from the `short` value after any markup commands have have locally redefined using `\glsxtrassignactualsetup`. The actual short value may then be modified by these attributes. Similarly, for the "actual long value".

Finally, if `shortaccess` hasn't already been set, it will be set to:

`\glsdefaultshortaccess{`⟨*actual long*⟩`}{`⟨*actual short*⟩`}`

(with `\glsdefaultshortaccess` expanded). This command is provided by glossaries−accsupp and is defined to do just {⟨*actual long*⟩}. It was redefined by glossaries−extra v1.42 to do {⟨*actual long*⟩} ({⟨*actual short*⟩}), but has been reverted back to its original definition in v1.49.

accessinsertdots=⟨*boolean*⟩

If the `shortaccess` key hasn't been set then this attribute will be checked. If true, the actual short value will have dots inserted (as per `insertdots`). Note that if this attribute hasn't been set but `insertdots` is true (and the `shortaccess` key hasn't been set), then the actual short value will also have dots inserted.

accessaposplural=⟨*boolean*⟩

If the `shortpluralaccess` key hasn't been set then this attribute will be checked. If true, the actual short plural value will have the apostrophe suffix (similar to `aposplural` but using `\glsxtrabbrvpluralsuffix` instead of `\abbrvpluralsuffix`). Note that if this attribute hasn't been set but `aposplural` is true (and the `shortpluralaccess` key hasn't been set), then the actual short plural value will also have the apostrophe suffix.

accessnoshortplural=⟨*boolean*⟩

If the `shortpluralaccess` key hasn't been set and the `accessaposplural` attribute hasn't been set, then this attribute will be checked. If true, the actual short plural value will be the same as the singular (as `noshortplural`). Note that if this attribute hasn't been set but `noshortplural` is true (and the `shortpluralaccess` key hasn't been set), then the actual short plural value will also be the singular form.

nameshortaccess=⟨*boolean*⟩

If the `access` key hasn't been set and this attribute is true, then the `access` field will be set to the same as the `shortaccess`.

textshortaccess=⟨*boolean*⟩

If the textaccess key hasn't been set and this attribute is true, then the textaccess field will be set to the same as the shortaccess. Additionally, if the pluralaccess key hasn't been set, then it will be set to the same as the shortpluralaccess value.

firstshortaccess=⟨*boolean*⟩

If the firstaccess key hasn't been set and this attribute is true, then the firstaccess field will be set to the same as the shortaccess. Additionally, if the firstpluralaccess key hasn't been set, then it will be set to the same as the shortpluralaccess value.

9.2. Accessibility Wrappers

The glossary style commands such as \glossentryname incorporate accessibility support by using the \glsaccess⟨*field*⟩ commands instead of the corresponding \glsentry⟨*field*⟩ commands.

If the glossaries-accsupp package hasn't been loaded or if the relevant accessibility field hasn't been set, these commands simply do the corresponding \glsentry⟨*field*⟩ command.

\glsaccessname{⟨*entry-label*⟩}

This shows the name field encapsulated with \glsnameaccessdisplay or just \glsentryname{⟨*entry-label*⟩}.

\Glsaccessname{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessname{⟨*entry-label*⟩}

As above but all caps.

\glsaccesstext{⟨*entry-label*⟩}

This shows the text field encapsulated with \glstextaccessdisplay or just \glsentrytext{⟨*entry-label*⟩}.

\Glsaccesstext{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccesstext{⟨*entry-label*⟩}

As above but all caps.

\glsaccessplural{⟨*entry-label*⟩}

This shows the `plural` field encapsulated with `\glspluralaccessdisplay` or just `\glsentryplural{⟨entry-label⟩}`.

\Glsaccessplural{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessplural{⟨*entry-label*⟩}

As above but all caps.

\glsaccessfirst{⟨*entry-label*⟩}

This shows the `first` field encapsulated with `\glsfirstaccessdisplay` or just `\glsentryfirst{⟨entry-label⟩}`.

\Glsaccessfirst{⟨*entry-label*⟩}

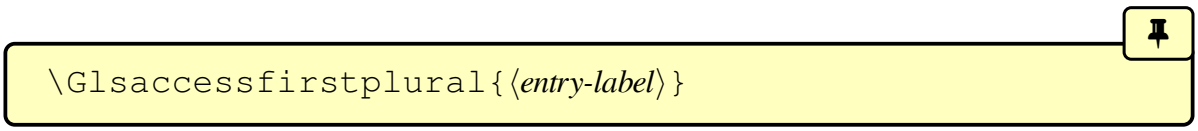
As above but sentence case.

\GLSaccessfirst{⟨*entry-label*⟩}

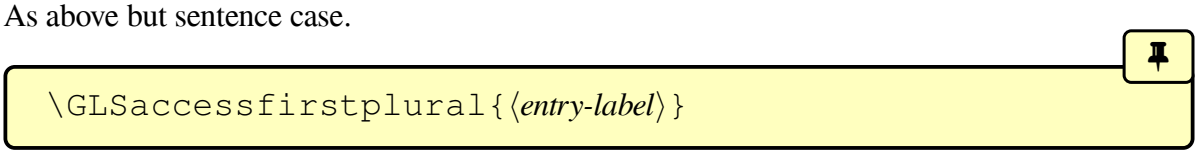
As above but all caps.

\glsaccessfirstplural{⟨*entry-label*⟩}

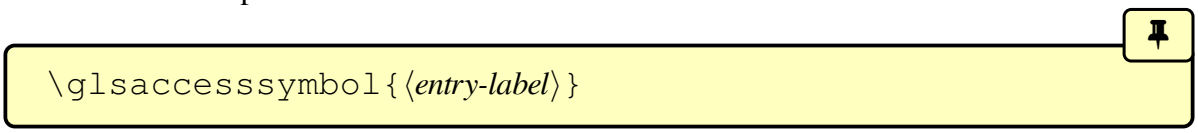
This shows the `firstplural` field encapsulated with `\glsfirstpluralaccess-display` or just `\glsentryfirstplural{⟨entry-label⟩}`.



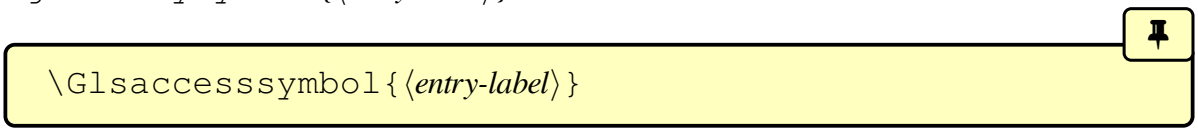
As above but sentence case.



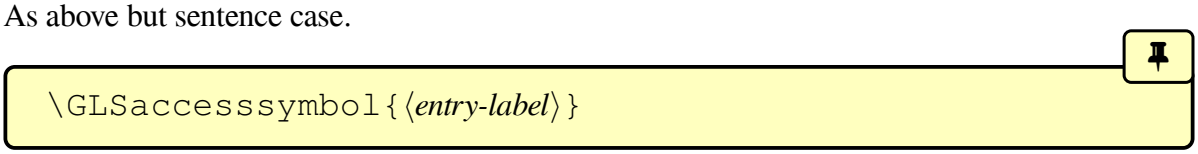
As above but all caps.



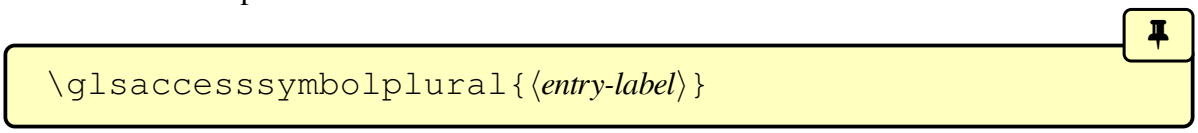
This shows the `symbol` field encapsulated with `\glssymbolaccessdisplay` or just `\glsentrysymbol{⟨entry-label⟩}`.



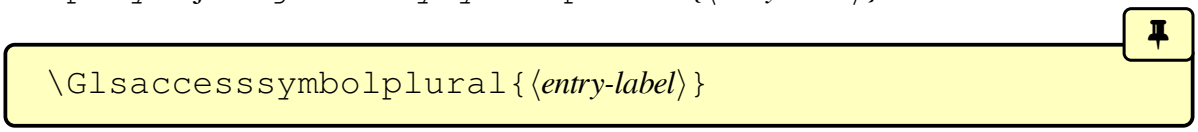
As above but sentence case.



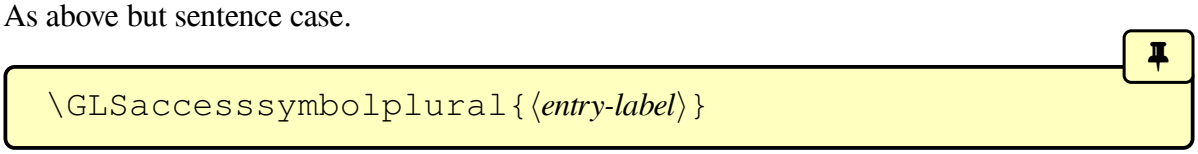
As above but all caps.



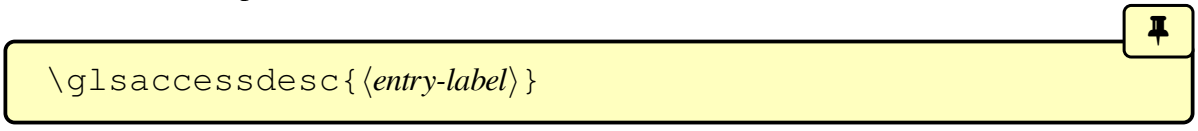
This shows the `symbolplural` field encapsulated with `\glssymbolpluralaccessdisplay` or just `\glsentrysymbolplural{⟨entry-label⟩}`.



As above but sentence case.



As above but all caps.



This shows the `description` field encapsulated with `\glsdescriptionaccessdisplay` or just `\glsentrydesc{⟨entry-label⟩}`.

\Glsaccessdesc{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessdesc{⟨*entry-label*⟩}

As above but all caps.

\glsaccessdescplural{⟨*entry-label*⟩}

This shows the descriptionplural field encapsulated with \glsdescriptionpluralaccessdisplay or just \glsentrydescplural{⟨*entry-label*⟩}.

\Glsaccessdescplural{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessdescplural{⟨*entry-label*⟩}

As above but all caps.

\glsaccessshort{⟨*entry-label*⟩}

This shows the short field encapsulated with \glsshortaccessdisplay or just \glsentryshort{⟨*entry-label*⟩}.

\Glsaccessshort{⟨*entry-label*⟩}

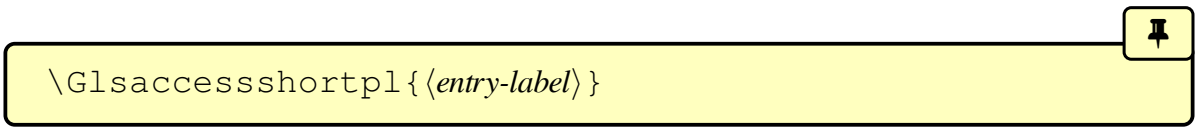
As above but sentence case.

\GLSaccessshort{⟨*entry-label*⟩}

As above but all caps.

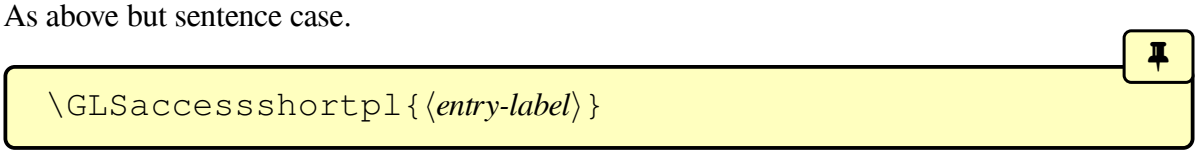
\glsaccessshortpl{⟨*entry-label*⟩}

This shows the shortplural field encapsulated with \glsshortpluralaccessdisplay or just \glsentryshortpl{⟨*entry-label*⟩}.



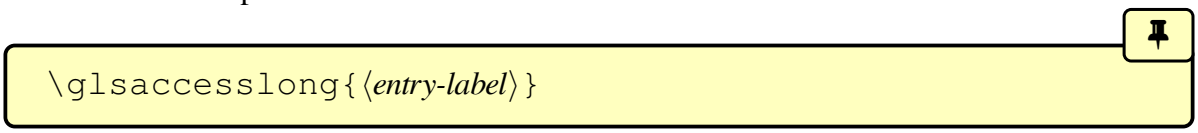
\Glsaccessshortpl{⟨*entry-label*⟩}

As above but sentence case.



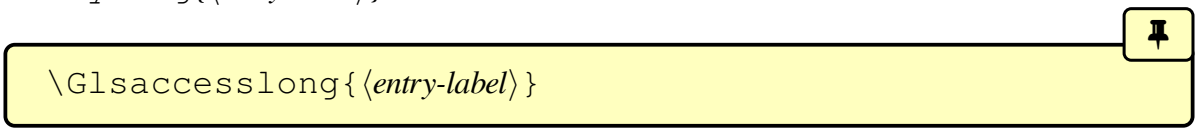
\GLSaccessshortpl{⟨*entry-label*⟩}

As above but all caps.



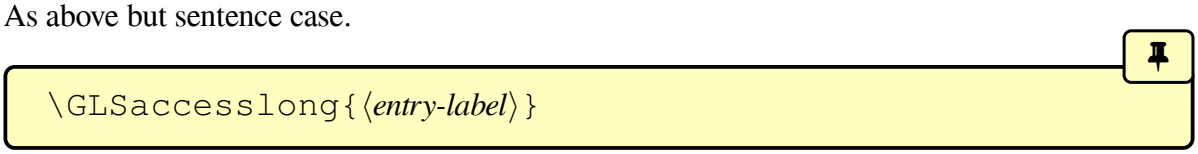
\glsaccesslong{⟨*entry-label*⟩}

This shows the `long` field encapsulated with `\glslongaccessdisplay` or just `\glsentrylong{`⟨*entry-label*⟩`}`.



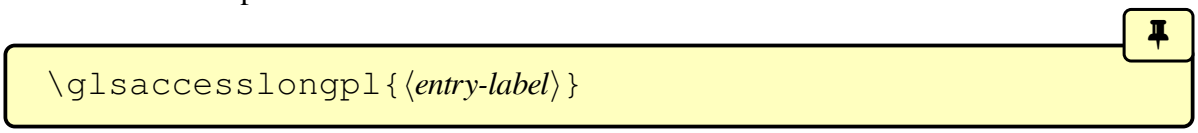
\Glsaccesslong{⟨*entry-label*⟩}

As above but sentence case.



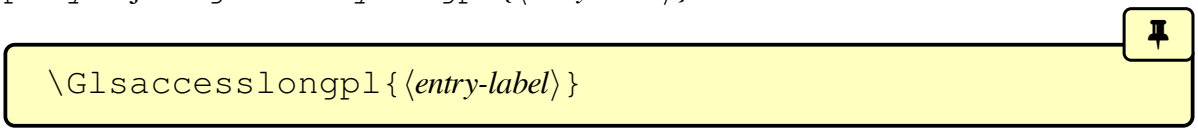
\GLSaccesslong{⟨*entry-label*⟩}

As above but all caps.



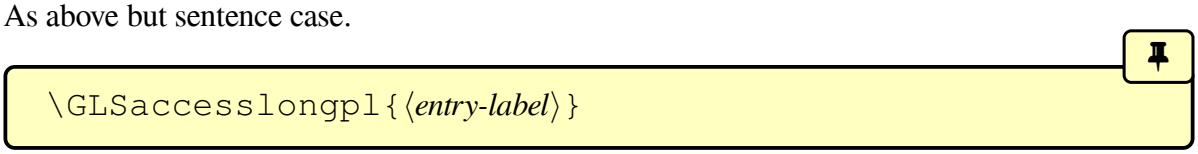
\glsaccesslongpl{⟨*entry-label*⟩}

This shows the `longplural` field encapsulated with `\glslongpluralaccessdisplay` or just `\glsentrylongpl{`⟨*entry-label*⟩`}`.



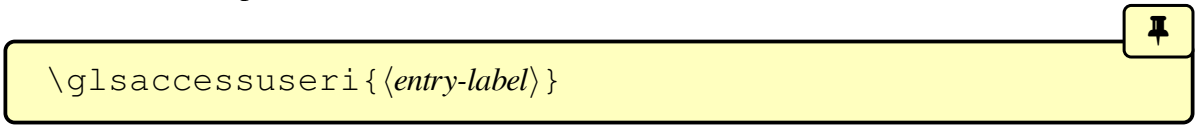
\Glsaccesslongpl{⟨*entry-label*⟩}

As above but sentence case.



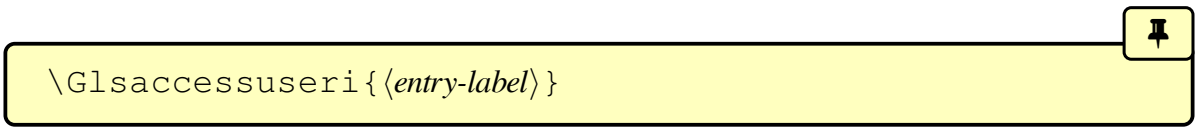
\GLSaccesslongpl{⟨*entry-label*⟩}

As above but all caps.

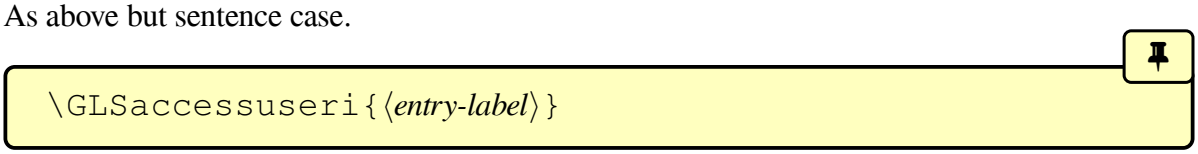


\glsaccessuseri{⟨*entry-label*⟩}

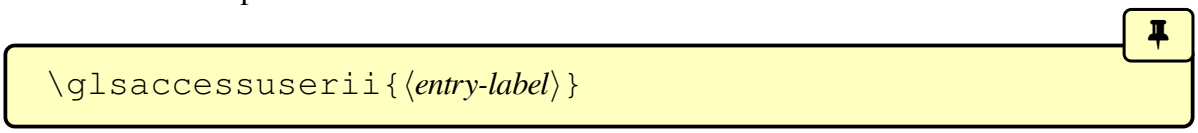
This shows the `user1` field encapsulated with `\glsuseriaccessdisplay` or just `\glsentryuseri{`⟨*entry-label*⟩`}`.



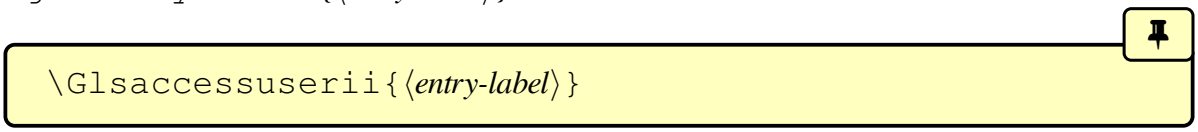
As above but sentence case.



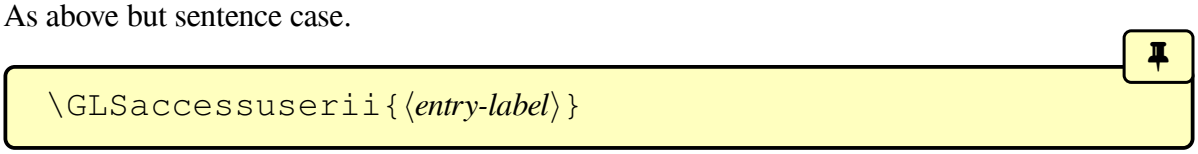
As above but all caps.



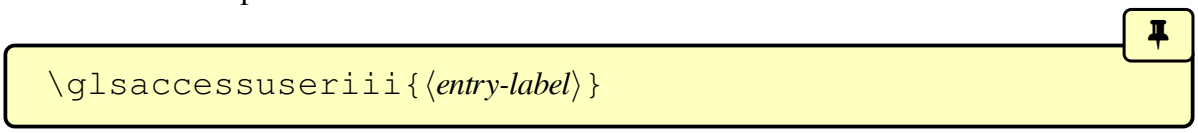
This shows the `user2` field encapsulated with `\glsuseriiaccessdisplay` or just `\glsentryuserii{⟨entry-label⟩}`.



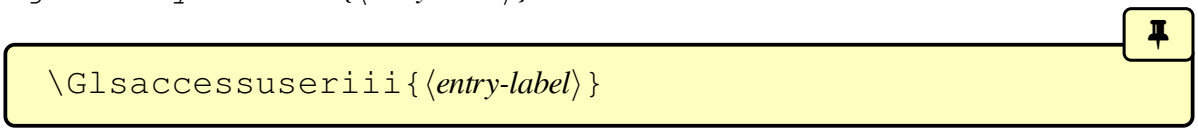
As above but sentence case.



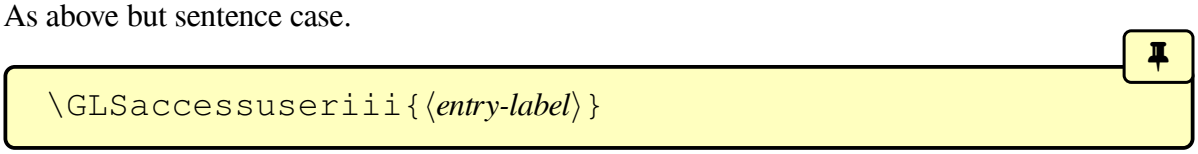
As above but all caps.



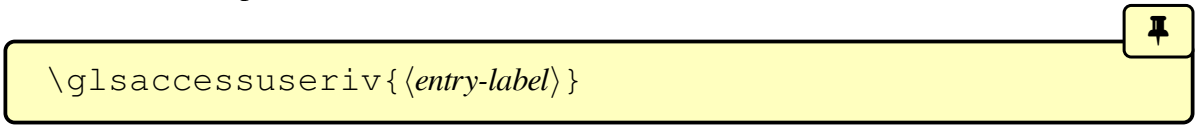
This shows the `user3` field encapsulated with `\glsuseriiiaccessdisplay` or just `\glsentryuseriii{⟨entry-label⟩}`.



As above but sentence case.



As above but all caps.



This shows the `user4` field encapsulated with `\glsuserivaccessdisplay` or just `\glsentryuseriv{⟨entry-label⟩}`.

\Glsaccessuseriv{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessuseriv{⟨*entry-label*⟩}

As above but all caps.

\glsaccessuserv{⟨*entry-label*⟩}

This shows the user5 field encapsulated with \glsuservaccessdisplay or just \glsentryuserv{⟨*entry-label*⟩}.

\Glsaccessuserv{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessuserv{⟨*entry-label*⟩}

As above but all caps.

\glsaccessuservi{⟨*entry-label*⟩}

This shows the user6 field encapsulated with \glsuserviaccessdisplay or just \glsentryuservi{⟨*entry-label*⟩}.

\Glsaccessuservi{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessuservi{⟨*entry-label*⟩}

As above but all caps.

9.3. Inner Formatting Wrappers

These \glsaccessfmt⟨*field*⟩ commands, such as \glsaccessfmtname, are similar to the corresponding \glsaccess⟨*field*⟩ commands, such as \glsaccessname, described above, but they format the field value using \glsfmtfield, \Glsfmtfield or \GLSfmtfield with the supplied ⟨*cs*⟩ encapsulating command.

The default entry display style \glsgenentryfmt, and the predefined abbreviation styles all incorporate accessibility support by using these commands in order to support the inner formatting.

\glsaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

This shows the name field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with \glsnameaccessdisplay.

\Glsaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

As above but sentence case.

\GLSaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

As above but all caps.

\glsaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

This shows the text field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with \glstextaccessdisplay.

\Glsaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

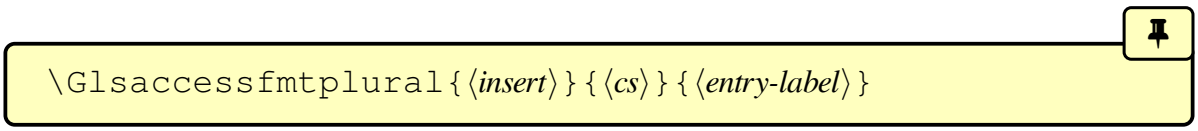
As above but sentence case.

\GLSaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

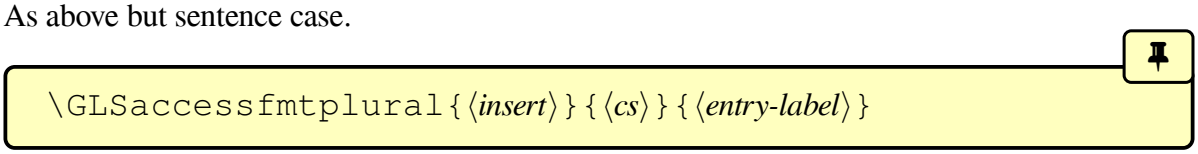
As above but all caps.

\glsaccessfmtplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}

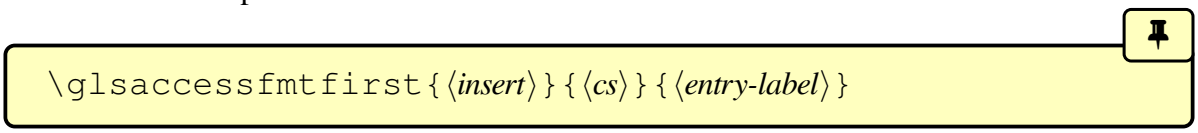
This shows the plural field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with \glspluralaccessdisplay.



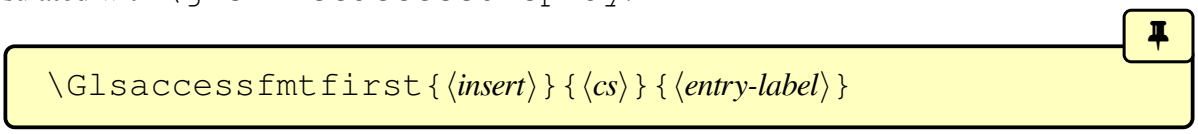
As above but sentence case.



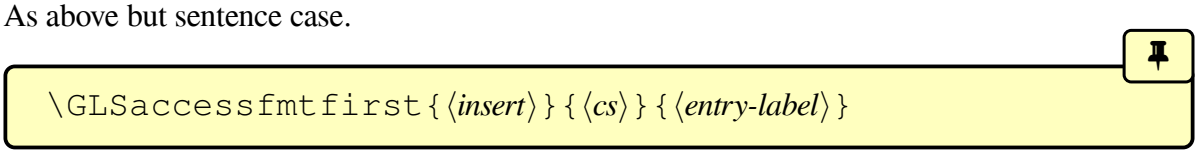
As above but all caps.



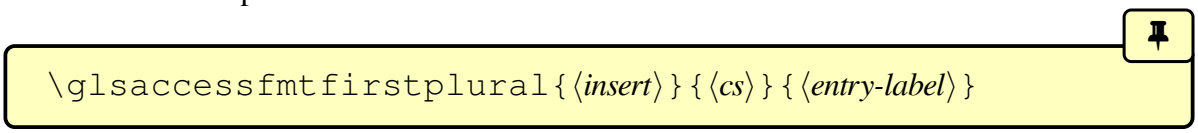
This shows the `first` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsfirstaccessdisplay`.



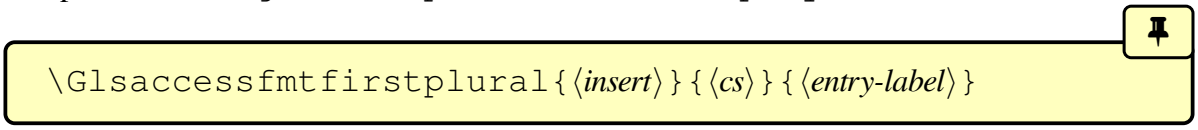
As above but sentence case.



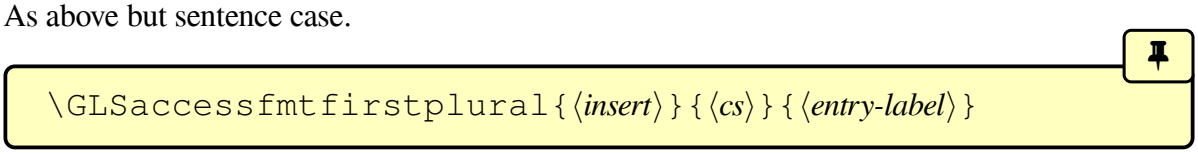
As above but all caps.



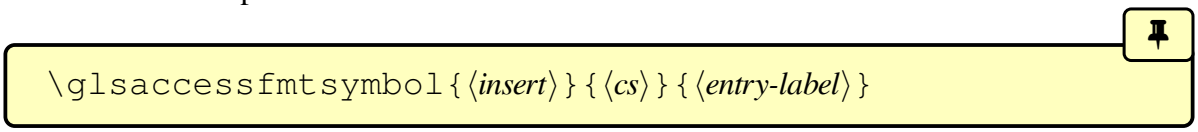
This shows the `firstplural` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsfirstpluralaccessdisplay`.



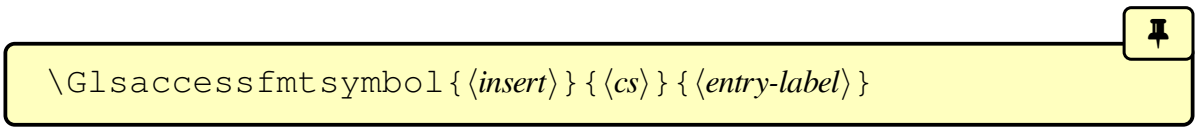
As above but sentence case.



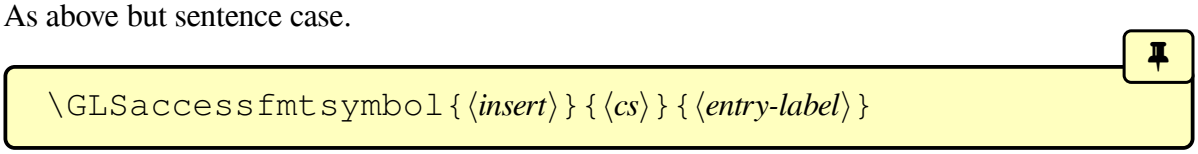
As above but all caps.



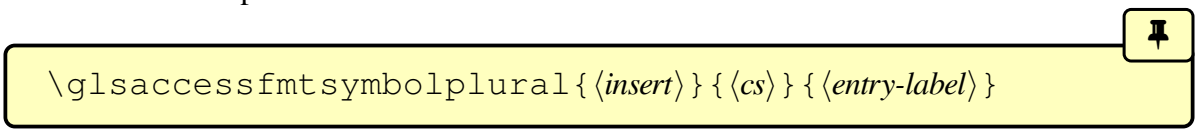
This shows the `symbol` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glssymbolaccessdisplay`.



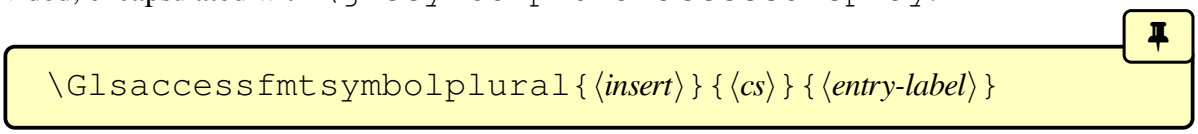
As above but sentence case.



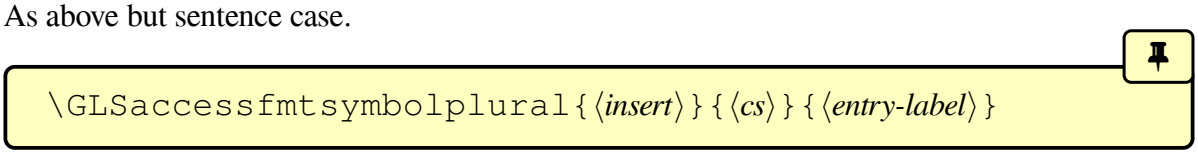
As above but all caps.



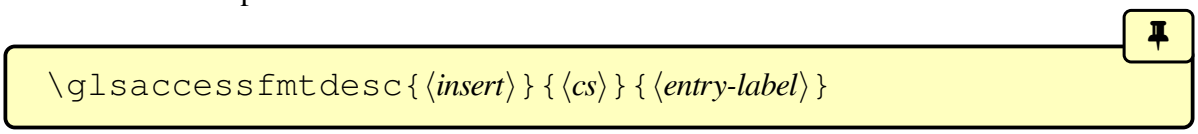
This shows the `symbolplural` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glssymbolpluralaccessdisplay`.



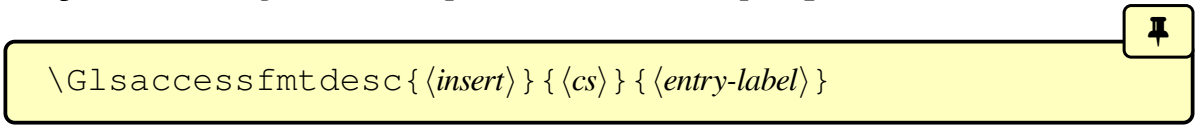
As above but sentence case.



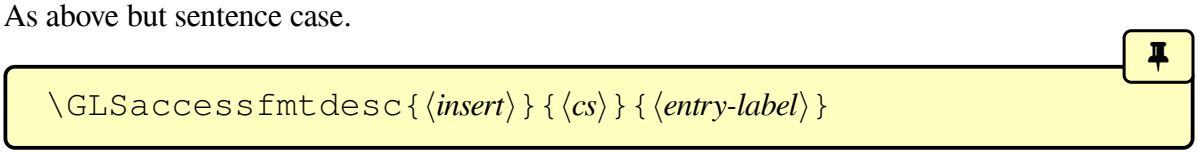
As above but all caps.



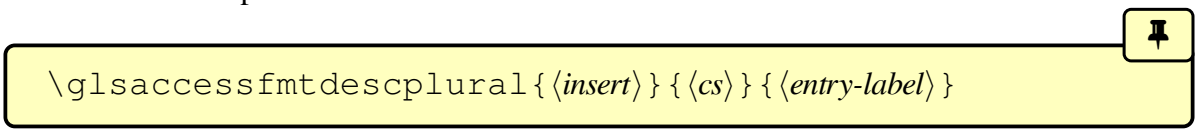
This shows the `description` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsdescriptionaccessdisplay`.



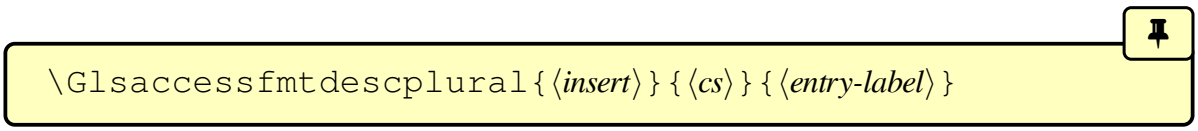
As above but sentence case.



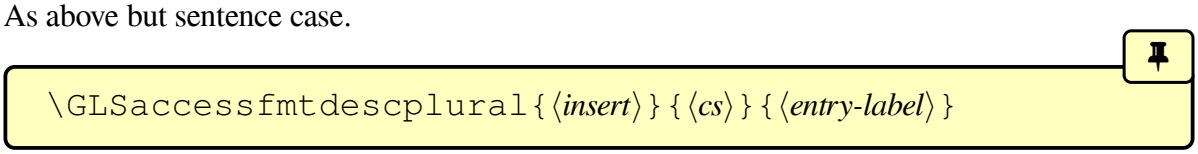
As above but all caps.



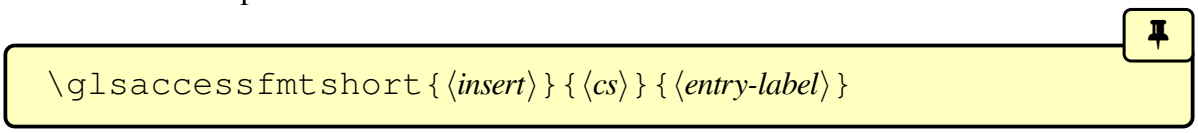
This shows the `descriptionplural` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsdescriptionpluralaccessdisplay`.



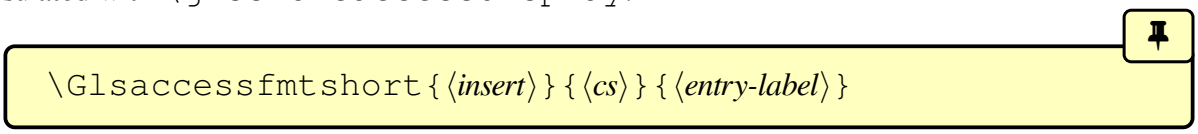
As above but sentence case.



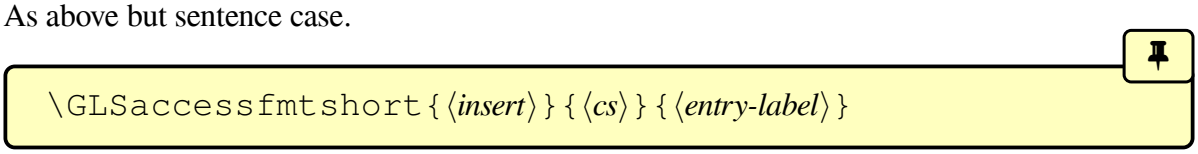
As above but all caps.



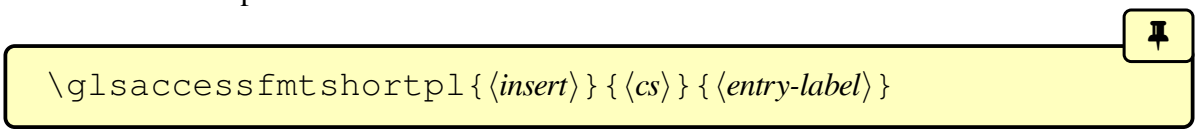
This shows the `short` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsshortaccessdisplay`.



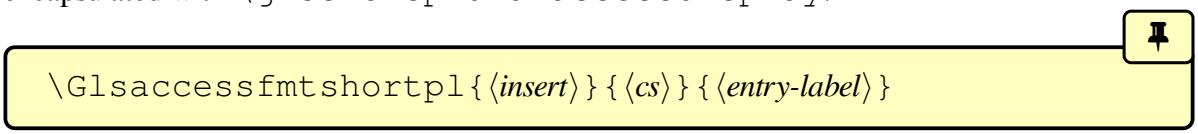
As above but sentence case.



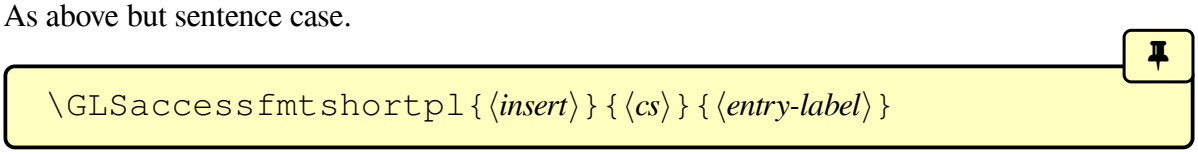
As above but all caps.



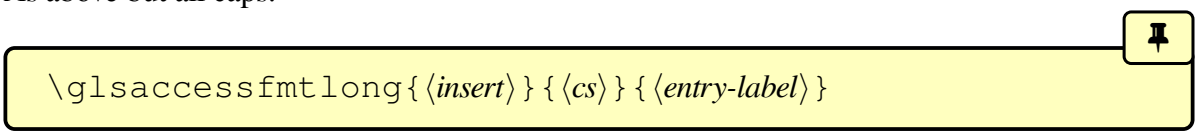
This shows the `shortplural` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsshortpluralaccessdisplay`.



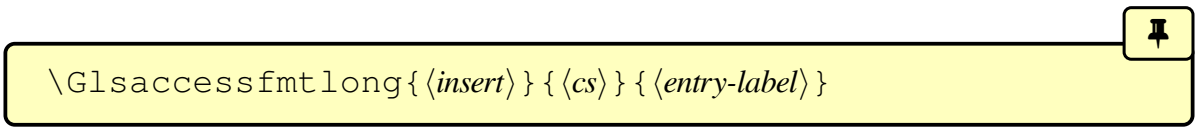
As above but sentence case.



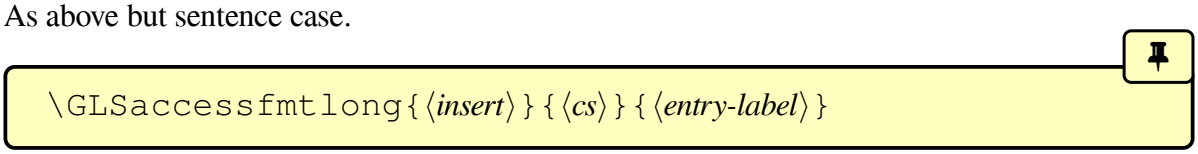
As above but all caps.



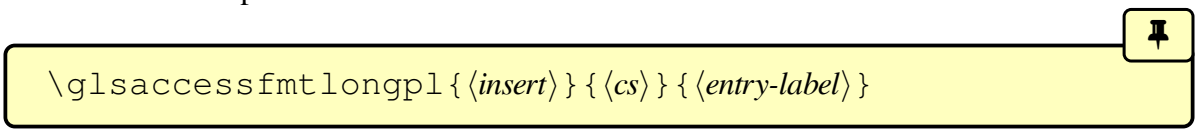
This shows the `long` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glslongaccessdisplay`.



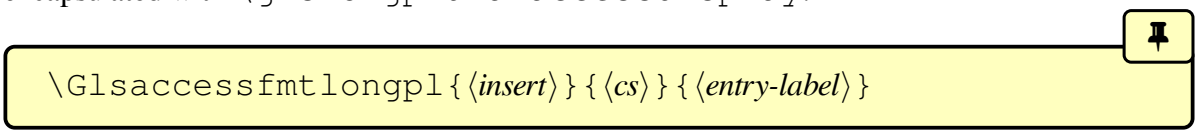
As above but sentence case.



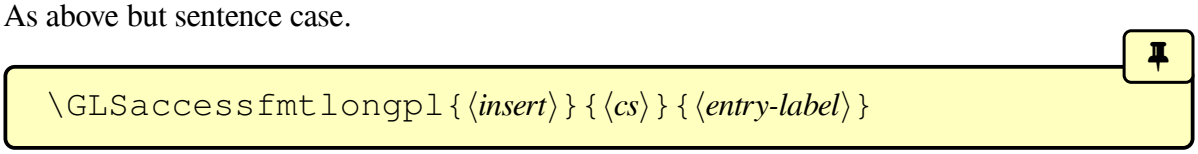
As above but all caps.



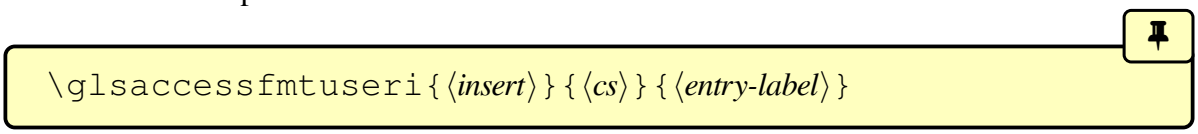
This shows the `longplural` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glslongpluralaccessdisplay`.



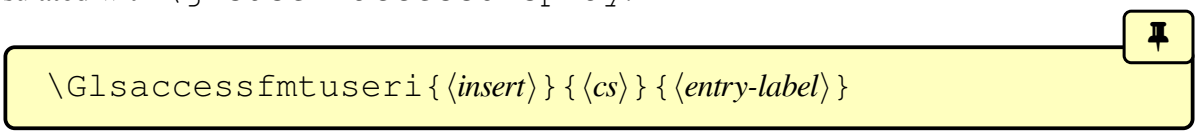
As above but sentence case.



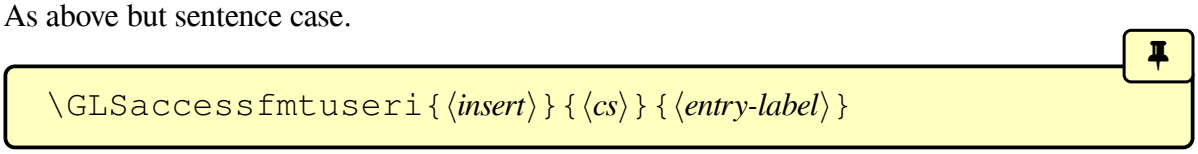
As above but all caps.



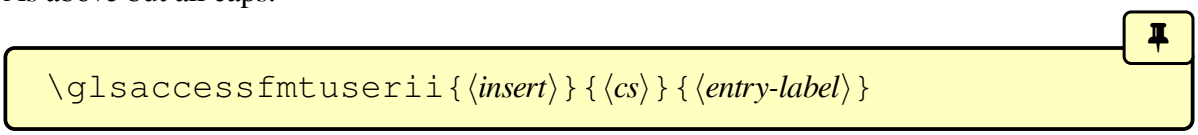
This shows the `user1` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuseriaccessdisplay`.



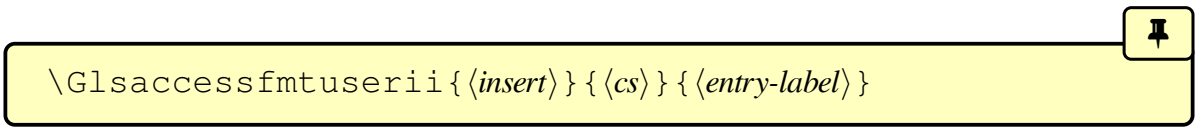
As above but sentence case.



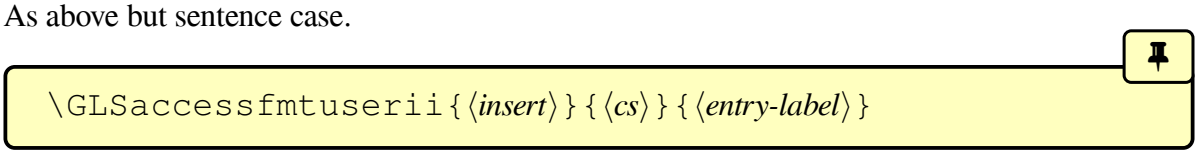
As above but all caps.



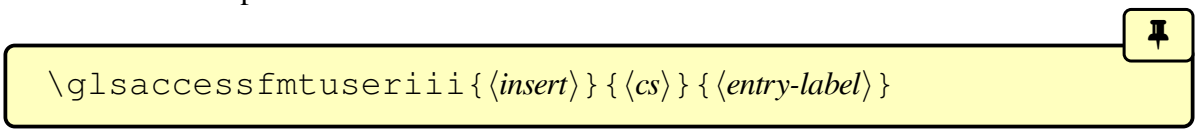
This shows the `user2` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuseriiaccessdisplay`.



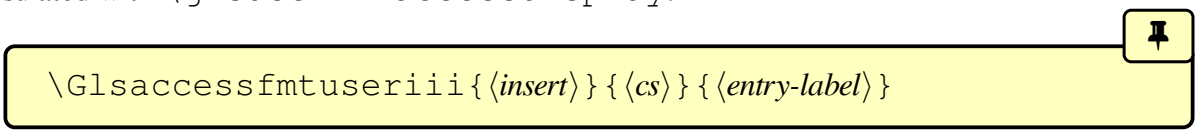
As above but sentence case.



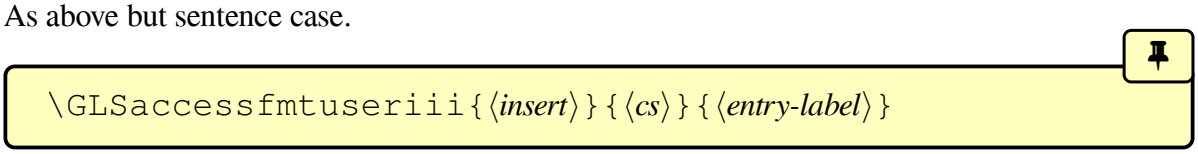
As above but all caps.



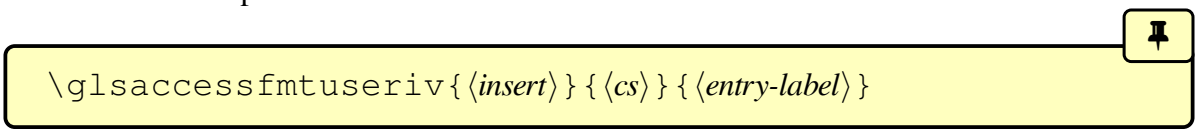
This shows the `user3` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuseriiiaccessdisplay`.



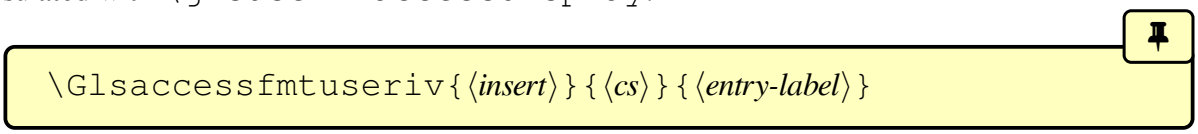
As above but sentence case.



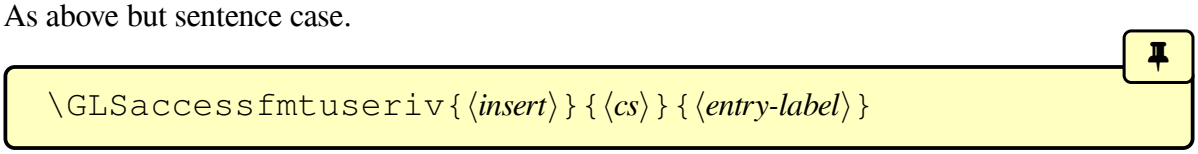
As above but all caps.



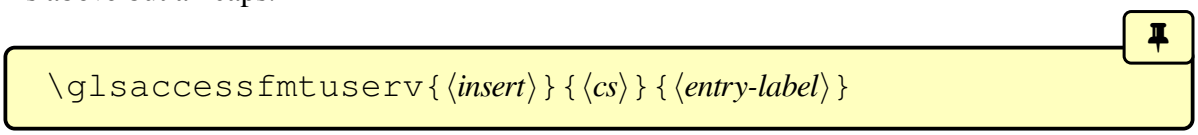
This shows the `user4` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuserivaccessdisplay`.



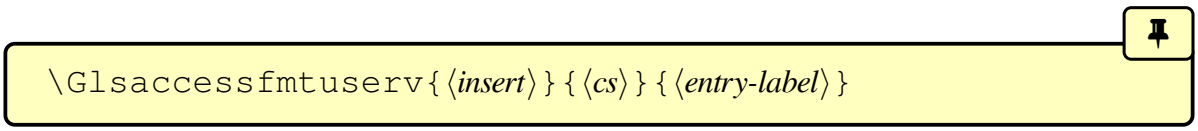
As above but sentence case.



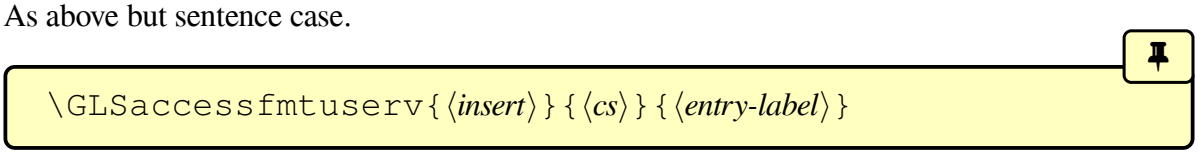
As above but all caps.



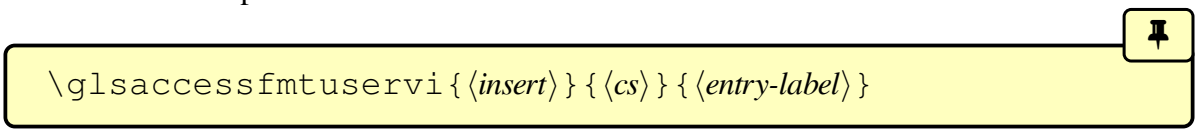
This shows the `user5` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuservaccessdisplay`.



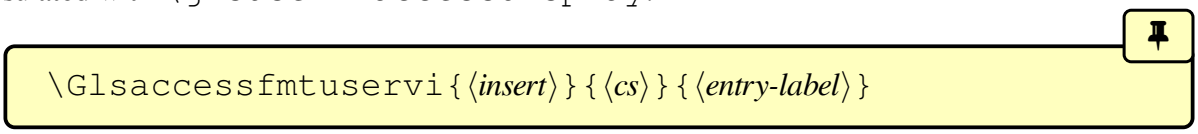
As above but sentence case.



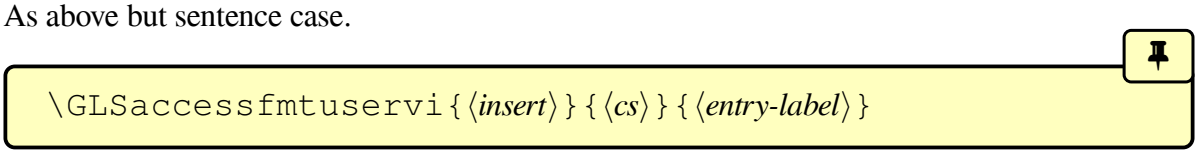
As above but all caps.



This shows the `user6` field formatted with ⟨*cs*⟩ and, if accessibility support provided, encapsulated with `\glsuserviaccessdisplay`.



As above but sentence case.



As above but all caps.

10. Categories

For multi-entry categories, see §7.8.

Each entry defined by \newglossaryentry (or commands that internally use it such as \newabbreviation) is assigned a category through the category key. You may add any category that you like, but since the category is a label used in the creation of some control sequences, avoid problematic characters within the category label. (So take care if you have babel shorthands on that make some characters active.)

The use of categories can give you more control over the way entries are displayed in the text or glossary. Note that an entry's category is independent of the glossary type. Be careful not to confuse category with type.

\glscategory{⟨*entry-label*⟩}

Expands to the category of the given entry or does nothing if the entry doesn't exist (analogous to \glsentryname).

\glsifcategory{⟨*entry-label*⟩}{⟨*category*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if the entry given by ⟨*entry-label*⟩ has the category set to ⟨*category*⟩.

An entry may have its category field changed using commands such as \GlsXtrSetField (see §3.5). In addition, the following commands are provided to batch set the category for a collection of entries.

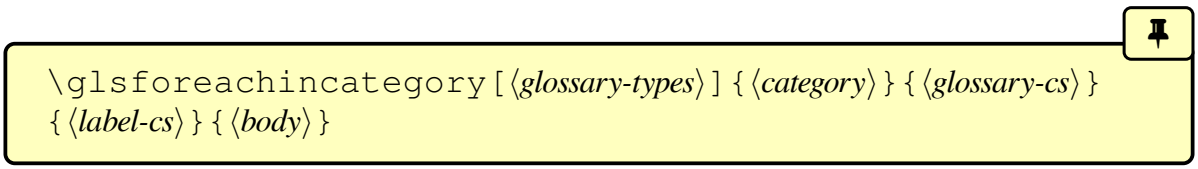
\glsxtrsetcategory{⟨*entry-labels*⟩}{⟨*category-label*⟩}

Globally sets the category field to the fully expanded ⟨*category-label*⟩ for each entry listed in ⟨*entry-labels*⟩.

\glsxtrsetcategoryforall{⟨*glossary-labels*⟩}{⟨*category-label*⟩}

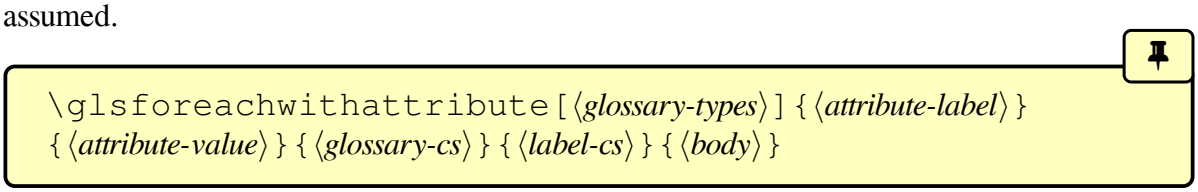
Globally sets the category field to the fully expanded ⟨*category-label*⟩ for each entry belonging to the glossaries listed in ⟨*glossary-labels*⟩.

There are also some iterative commands available:



\glsforeachincategory[⟨glossary-types⟩]{⟨category⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}

This iterates through all entries in the glossaries identified by the comma-separated list ⟨*glossary-labels*⟩ that have the category given by ⟨*category-label*⟩ and performs ⟨*body*⟩ for each match. Within ⟨*body*⟩, you can use ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ (which much be control sequences) to access the current glossary and entry label. If ⟨*glossary-labels*⟩ is omitted, all glossaries are assumed.

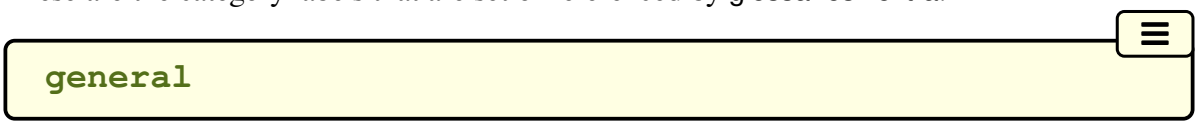


\glsforeachwithattribute[⟨glossary-types⟩]{⟨attribute-label⟩}{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}

This iterates over all entries in the given list of glossaries that have a category with the given ⟨*attribute-label*⟩ set to ⟨*attribute-value*⟩ and performs ⟨*body*⟩ at each iteration. If ⟨*glossary-types*⟩ is omitted, the list of all non-ignored glossaries is assumed. The remaining arguments are as for `\glsforeachincategory`.

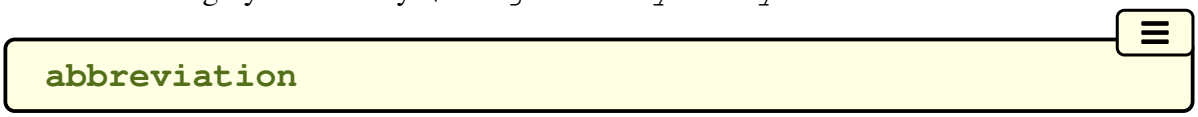
10.1. Known Categories

These are the category labels that are set or referenced by glossaries−extra.



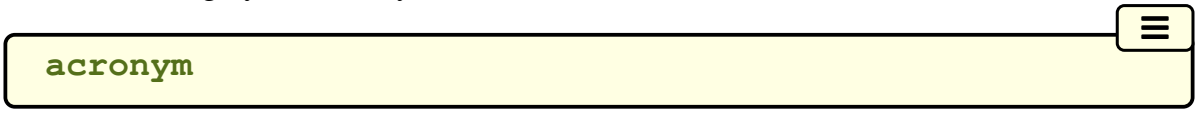
general

The default category assumed by `\newglossaryentry`.



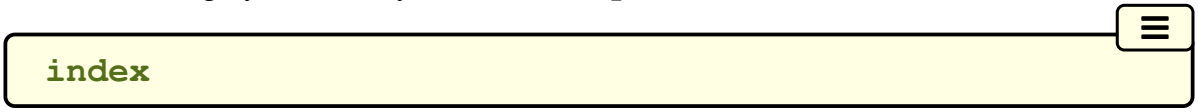
abbreviation

The default category assumed by `\newabbreviation`.



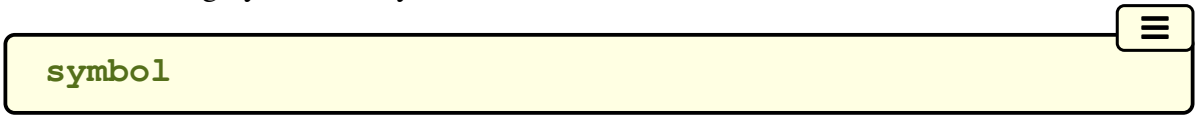
acronym

The default category assumed by `\newacronym`.



index

The default category assumed by `\newterm`.



symbol

The default category assumed by `\glsxtrnewsymbol`.

number

The default category assumed by `\glsxtrnewnumber`.

10.2. Attributes

Each category may have a set of attributes, where each attribute has an associated value for its given category. An entry's attribute set corresponds to the attributes associated with the entry's category.

As with the category, the attribute name is also a label. You can provide your own custom attributes, which you can set and access with the commands described in §10.2.2.

10.2.1. Known Attributes

This section lists attributes that glossaries−extra sets or accesses. If an attribute hasn't been set, a default is assumed. For boolean attributes, the test may simply be to determine if the attribute has been set to `true`, in which case any other value or a missing value will be interpreted as false. Conversely, the test may be to determine if the attribute has been set to `false`, in which case any other value or a missing value will be interpreted as true.

See §7.8 for attributes relating to multi-entry categories.

10.2.1.1. Abbreviation Attributes

See §9.1 for abbreviation accessibility attributes.

regular=⟨*boolean*⟩

This attribute indicates whether or not an entry should be considered a regular entry. This enables `\glsentryfmt` to determine whether to use `\glsgenentryfmt` or `\glsxtrgenabbrvfmt`.

The `general` and `acronym` categories have the `regular` attribute automatically set to `true`. Some abbreviation styles change this value.

discardperiod=⟨*boolean*⟩

If set to "true", the post-link hook will discard a full stop that follows *non-plural* commands like `\gls` or `\glstext`(see §5.5.4).

This attribute doesn't apply to the accessibility fields. See §9.1 for attributes related to accessibility support for abbreviations.

Note that this can cause a problem if you access a field that doesn't end with a full stop. For example:

```
\newabbreviation
 [user1={German Speaking \TeX\ User Group}]
 {dante}{DANTE e.V.}{Deutschsprachige
  Anwendervereinigung \TeX\ e.V.}
```

Here the `short` and `long` fields end with a full stop, but the `user1` field doesn't. The simplest solution in this situation is to put the sentence terminator in the final optional argument. For example:

```
\glsuseri{dante}[.]
```

This will bring the punctuation character inside the link text and it won't be discarded.

pluraldiscardperiod=⟨*boolean*⟩

If this attribute is set to "true" *and* the `discardperiod` attribute is set to "true", this will behave as above for the plural commands like `\glspl` or `\glsplural`.

retainfirstuseperiod=⟨*boolean*⟩

If this attribute is set to "true" then the discard is determined by `\glsxtrdiscardperiodretainfirstuse`, regardless of the `discardperiod` or `pluraldiscardperiod` attributes. This is useful for ⟨*short*⟩ (⟨*long*⟩) abbreviation styles where only the short form has a trailing full stop.

This attribute doesn't apply to the accessibility fields. See §9.1 for attributes related to accessibility support for abbreviations.

markwords=⟨*boolean*⟩

If this attribute is set to "true" any entry defined using `\newabbreviation` will automat-

ically have spaces in the long form replaced with:

```
\glsxtrwordsep
```

and each word is encapsulated with:

```
\glsxtrword{⟨word⟩}
```

For example:

```
\glssetcategoryattribute{abbreviation}{markwords}
{true}
\newabbreviation{ip}{IP}{Internet Protocol}
```

is essentially the same as

```
\newabbreviation{ip}{IP}
{\glsxtrword{Internet}\glsxtrwordsep\glsxtrword
{Protocol}}
```

The "hyphen" styles, such as long−hyphen−short−hyphen, take advantage of this markup. If the inserted material (provided in the final argument of `\gls`-like commands) starts with a hyphen then `\glsxtrwordsep` is locally redefined to a hyphen. (The default value is a space). Note that this only applies to commands like `\gls` and not like `\glsxtrlong`. You can provide your own localised switch, if required. For example:

```
\newcommand{\hyplong}[2][]{%
 {\def\glsxtrwordsep{-}\glsxtrlong[#1]{#2}}}
```

This setting will also adjust the long plural. This attribute is only applicable to entries defined using `\newabbreviation` (or `\newacronym` if it's using `\newabbreviation`.)

This setting may result in the `\glsxtrword` and `\glsxtrwordsep` markup ending up in the `sort` field, depending on the style in use.

markshortwords=⟨*boolean*⟩

This is similar to `markwords` but applies to the short form. (Only useful for abbreviations

that contain spaces.) This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it's using \newabbreviation.)

This setting will only adjust the short plural if the shortplural key isn't used. This setting will take precedence over insertdots.

This setting may result in the \glsxtrword and \glsxtrwordsep markup ending up in the sort field, depending on the style in use.

insertdots=⟨*boolean*⟩

If this attribute is set to "true" any entry defined using \newabbreviation will automatically have full stops inserted after each letter. The entry will be defined with those dots present as though they had been present in the ⟨*short*⟩ argument of \newabbreviation (rather than inserting them every time the entry is used). The short plural form defaults to the new dotted version of the original ⟨*short*⟩ form with the plural suffix appended. *This setting is incompatible with markshortwords.* This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it's using \newabbreviation.)

If you explicitly override the short plural using the shortplural key, you must explicitly insert the dots yourself (since there's no way for the code to determine if the plural has a suffix that shouldn't be followed by a dot).

This attribute is best used with the discardperiod attribute set to "true".

aposplural=⟨*boolean*⟩

If this attribute is set to "true", \newabbreviation will insert an apostrophe (') before the plural suffix for the *short* plural form (unless explicitly overridden with the shortplural key). The long plural form is unaffected by this setting. This setting overrides noshortplural. This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it's using \newabbreviation.) Check with your supervisor, publisher or editor if you want to use this attribute as this usage is controversial.

noshortplural=⟨*boolean*⟩

If this attribute is set to "true", \newabbreviation won't append the plural suffix for the short plural form. This means the short and shortplural values will be the same unless explicitly overridden. *This setting is incompatible with aposplural.* This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it's using \newabbreviation.)

tagging=⟨*boolean*⟩

If this attribute is set to "true", the tagging command defined by `\GlsXtrEnableInitialTagging` will be activated to use `\glsxtrtagfont` in the glossary (see §4.4)

10.2.1.2. Attributes that Alter `\glslink` Options

nohyperfirst=⟨*boolean*⟩

When used with the `\gls`-like commands, if this attribute is set to `true`, this will automatically suppress the hyperlink on first use.

This settings can be overridden by explicitly setting the `hyper` key on or off in the optional argument of the `\gls`-like command.

As from version 1.07, `\glsfirst`, `\Glsfirst`, `\GLSfirst` and their plural versions (which should ideally behave in a similar way to the first use of `\gls` or `\glspl`) now honour this attribute (but not the package-wide `hyperfirst=false` option, which matches the behaviour of glossaries). If you want commands like `\glsfirst` to ignore the `nohyperfirst` attribute then just redefine `\glsxtrchecknohyperfirst` to do nothing.

nohypernext=⟨*boolean*⟩

If set to `true`, this will automatically set `hyper=false` on subsequent use when using the `\gls`-like commands.

nohyper=⟨*boolean*⟩

If set to `true`, this will automatically set `hyper=false` when using the `\gls`-like or `\glstext`-like commands.

indexonlyfirst=⟨*boolean*⟩

This is similar to the `indexonlyfirst` package option but only for entries that have a category with this attribute set to "true".

wrgloss=⟨*value*⟩

When using the \gls-like or \glstext-like commands, this will automatically set wrgloss=after it this attribute is set to "after".

textformat=⟨*cs-name*⟩

The \gls-like and \glstext-like commands have the link text encapsulated in the argument of \glstextformat by default (the outer formatting, see §5.5.1). If the textformat attribute is set, the control sequence given by the attribute value will be used instead. The attribute value should be the name (without the leading backslash) of a command that takes a single argument (the link text). Remember that the abbreviation styles may apply an additional font change.

hyperoutside=⟨*boolean*⟩

This boolean attribute may be false, true or unset. If unset, true is assumed. This indicates the default setting of the hyperoutside option, described in §5.1.

10.2.1.3. Glossary Attributes

glossdesc=⟨*value*⟩

This attribute is checked by the \glossentrydesc to determine whether or not to apply any case change. The value may be one of:

firstuc

: Applies sentence case. That is, the first letter of the description will be converted to uppercase (using \Glsentrydesc).

title

: Applies title case. If you have at least glossaries v4.48, the title casing is indirectly performed by \glscapitalisewords, which defaults to \capitalisewords (provided by mfirstuc). You can either redefine \glscapitalisewords if you want the change to also affect \glsentrytitlecase or if you only want the change to apply to the attribute case-changing then redefine \glsxtrfieldtitlecasecs. For example:

```
\renewcommand*{\glsxtrfieldtitlecasecs}[1]
{\xcapitalisefmtwords*{#1}}
```

(Note that the argument to \glsxtrfieldtitlecasecs will be a control sequence whose replacement text is the entry's description, which is why \xcapitalisefmtwords is needed instead of \capitalisefmtwords.)

If an error occurs with this setting, try redefining \glsxtrfieldtitlecasecs as shown above.

Any other values of this attribute are ignored. Remember that there are design limitations for both the sentence case and the title case commands. See the mfirstuc user manual for further details.

If you are using bib2gls, you can use the description-case-change setting instead.

glossdescfont=⟨*cs-name*⟩

If set, the value should be the name of a control sequence (without the leading backslash) that takes one argument. This control sequence will be applied by \glossentrydesc to the description text. For example:

```
\glssetcategoryattribute{general}{glossdescfont}
{emph}
```

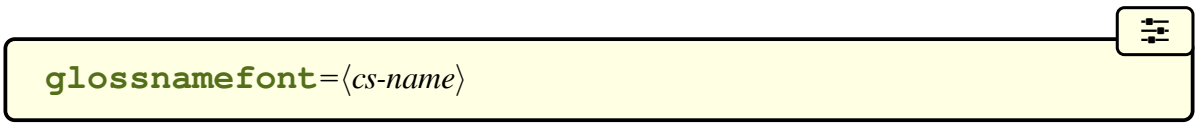
glossname=⟨*value*⟩

As glossdesc but applies to \glossentryname. Additionally, if this attribute is set to "uc" the name is converted to all caps.

If you are using bib2gls, you can use the name-case-change setting instead.

indexname={⟨*value*⟩}

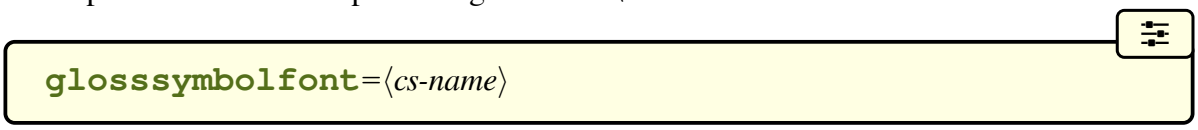
If set, the post-name hook will index the entry using \index. See §12 for further details.



glossnamefont=⟨*cs-name*⟩

As `glossdescfont` but applies to `\glossentryname`. Note that this overrides `\glsnamefont` which will only be used if this attribute hasn't been set.

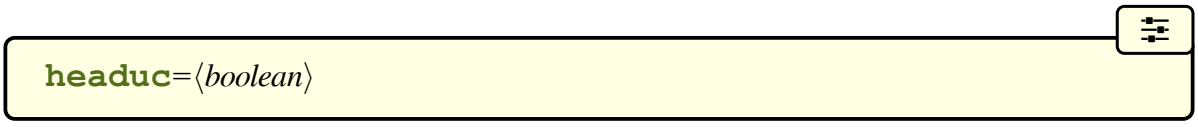
Remember that glossary styles may additionally apply a font change, such as the list styles which put the name in the optional argument of `\item`.



glosssymbolfont=⟨*cs-name*⟩

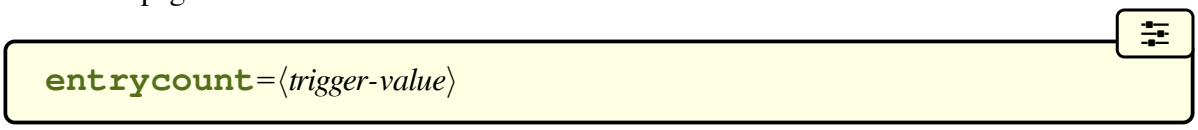
This is similar to `glossnamefont` and `glossdescfont` but is used by `\glossentrysymbol`.

10.2.1.4. Other Attributes



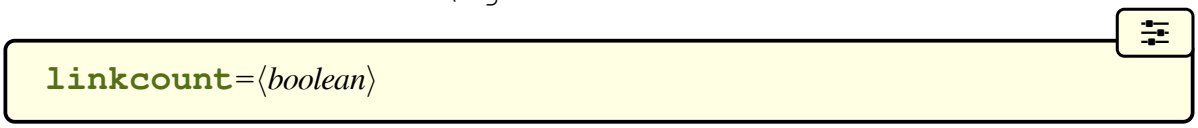
headuc=⟨*boolean*⟩

If this attribute is set to "true", commands like `\glsfmtshort` will use the upper case version in the page headers.



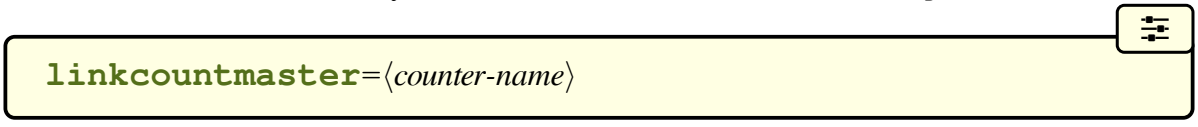
entrycount=⟨*trigger-value*⟩

The value of this attribute (if set) must be an integer and is used in combination with `\glsenableentrycount` (see §6.1). Leave blank or undefined for categories that shouldn't have this facility enabled. The value of this attribute is used by `\glsxtrifcounttrigger` to determine how commands such as `\cgls` should behave.



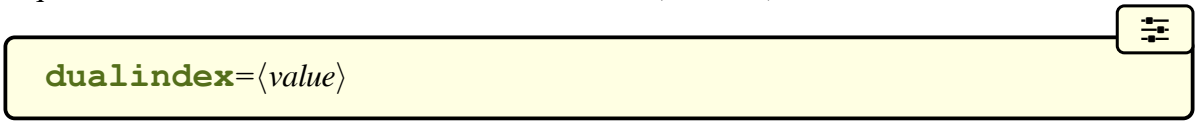
linkcount=⟨*boolean*⟩

This attribute is set to `true` by `\GlsXtrEnableLinkCounting` (see §6.2).



linkcountmaster=⟨*counter-name*⟩

This attribute is set by `\GlsXtrEnableLinkCounting` to the name of the counter that requires the link counter to be added to its reset list (see §6.2).



dualindex=⟨*value*⟩

If this attribute is set, whenever a glossary entry has information written to the external glossary file through commands like `\gls` and `\glsadd`, a corresponding line will be written to the

indexing file using \index. The value may be true to simply enable this feature or the value may be the encap to use with \index. See §12 for further details.

targeturl=⟨*url*⟩

If set, the hyperlink generated by commands like \gls will be set to the URL provided by this attribute's value. For example:

```
\glssetcategoryattribute
 {general}{targeturl}{master-doc.pdf}
```

(See also the accompanying sample file sample-external.tex.) If the URL contains awkward characters (such as % or ~) remember that the base glossaries package provides commands like \glspercentchar and \glstildechar that expand to literal characters.

targetname=⟨*anchor*⟩

If you want to a named anchor within the target URL (notionally adding #⟨*name*⟩ to the URL), then you also need to set targetname to the anchor ⟨*name*⟩. You may use \glslabel within ⟨*name*⟩ which is set by commands like \gls to the entry's label.

All the predefined glossary styles start each entry listing with \glstarget which sets the anchor to \glolinkprefix\glslabel, so if you want entries to link to glossaries in the URL given by targeturl, you can just do:

```
\glssetcategoryattribute{general}{targetname}
 {\glolinkprefix\glslabel}
```

(If the target document changed \glolinkprefix then you will need to adjust the above as appropriate.)

targetcategory=⟨*anchor*⟩

If the anchor is in the form ⟨*name1*⟩.⟨*name2*⟩ then use targetname for the ⟨*name2*⟩ part and targetcategory for the ⟨*name1*⟩ part.

For example:

```
\glssetcategoryattribute
 {general}{targeturl}{master-doc.pdf}
\glssetcategoryattribute
```

```
 {general}{targetcategory}{page}
\glssetcategoryattribute{general}{targetname}{7}
```

will cause all link text for `general` entries to link to `master-doc.pdf#page.7` (page 7 of that PDF).

If you want a mixture in your document of entries that link to an internal glossary and entries that link to an external URL then you can use `\newignoredglossary*` for the external list. For example:

```
\newignoredglossary*{external}

\glssetcategoryattribute{external}{targeturl}
{master-doc.pdf}
\glssetcategoryattribute
 {general}{targetname}{\glolinkprefix\glslabel}

\newglossaryentry{sample}{name={sample},description=
{local example}}

\newglossaryentry{sample2}{name={sample2},
  type={external},
  category={external},
  description={external example}}
```

externallocation=⟨*PDF filename*⟩

The value should be the file name of the target document when manually indexing an external location with `thevalue`. In general, it's better to use `bib2gls` v1.7+ which can handle multiple external sources and doesn't require this attribute.

10.2.2. Accessing and Setting Attributes

Attributes can be set using the following commands:

`\glssetcategoryattribute{`⟨*category*⟩`}{`⟨*attribute*⟩`}{`⟨*value*⟩`}`

Locally sets the given attribute to ⟨*value*⟩ for the given category.

`\glssetcategoriesattribute{`⟨*category list*⟩`}{`⟨*attribute*⟩`}{`⟨*value*⟩`}`

Globally sets the given attribute to ⟨*value*⟩ for all the categories in the comma-separated list ⟨*category list*⟩.

```
\glssetcategoryattributes{⟨category⟩}{⟨attribute list⟩}{⟨value⟩}
```

Globally sets each attribute in the comma separated ⟨*attribute list*⟩ to ⟨*value*⟩ for the given ⟨*category*⟩.

```
\glssetcategoriesattributes{⟨category list⟩}{⟨attribute
list⟩}{⟨value⟩}
```

Globally sets each attribute in the comma separated ⟨*attribute list*⟩ to ⟨*value*⟩ for each category in the comma-separated list ⟨*category list*⟩.

```
\glssetattribute{⟨entry-label⟩}{⟨attribute⟩}{⟨value⟩}
```

Locally sets the given attribute to ⟨*value*⟩ for the category associated with the entry identified by ⟨*entry-label*⟩. This command can't be used to assign an attribute for a multi-entry category.

```
\glssetregularcategory{⟨category⟩}
```

A shortcut that sets the `regular` attribute to `true` for the given category using `\glssetcategoryattribute`.

An attribute can be locally unset using:

```
\glsunsetcategoryattribute{⟨category⟩}{⟨attribute⟩}
```

Attribute values can be obtained with the following commands:

```
\glsgetcategoryattribute{⟨category⟩}{⟨attribute⟩}
```

Expands to the value of the given attribute for the given category. Expands to nothing if the attribute hasn't been set.

```
\glsgetattribute{⟨entry-label⟩}{⟨attribute⟩}
```

Expands to the value of the given attribute for the category associated with the entry identified by ⟨*entry-label*⟩. Expands to nothing if the attribute hasn't been set. This command can't be used to assign an attribute for a multi-entry category.

Attributes can be tested with the following commands.

```
\glshascategoryattribute{⟨category⟩}{⟨attribute⟩}{⟨true⟩}{⟨false⟩}
```

This uses etoolbox's `\ifcsvoid` and does ⟨*true*⟩ if the attribute has been set and isn't blank and isn't `\relax` otherwise it does ⟨*false*⟩.

```
\glshasattribute{⟨entry-label⟩}{⟨attribute⟩}{⟨true⟩}{⟨false⟩}
```

As `\glshascategoryattribute` but the category is obtained from the given entry. This command can't be used to test an attribute associated with a multi-entry category.

```
\glsifcategoryattribute{⟨category⟩}{⟨attribute⟩}{⟨value⟩}{⟨true⟩}
{⟨false⟩}
```

This tests if the given attribute for the given category is set and equal to ⟨*value*⟩. If true, ⟨*true*⟩ is done. If the attribute isn't set or is set but isn't equal to ⟨*value*⟩, ⟨*false*⟩ is done.

For example:

```
\glsifcategoryattribute{general}{nohyper}{true}
{NO HYPER}{HYPER}
```

This does "NO HYPER" if the `general` category has the `nohyper` attribute set to `true` otherwise if does "HYPER".

```
\glsifattribute{⟨entry-label⟩}{⟨attribute⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}
```

As `\glsifcategoryattribute` but the category is obtained from the given entry. This command can't be used to test an attribute associated with a multi-entry category.

```
\glsifregularcategory{⟨category⟩}{⟨true⟩}{⟨false⟩}
```

A shortcut that tests if the given category has the `regular` attribute set to `true`.

```
\glsifnotregularcategory{⟨category⟩}{⟨true⟩}{⟨false⟩}
```

A shortcut that tests if the given category has the `regular` attribute set to `false`.

If the regular attribute hasn't been set, both \glsifregularcategory and \glsifnotregularcategory will do ⟨*false*⟩. The choice of command needs to be determined by what outcome should occur if the attribute hasn't been set.

\glsifregular{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}

As \glsifregularcategory but the category is obtained from the given entry. This command can't be used to test an attribute associated with a multi-entry category.

\glsifnotregular{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}

As \glsifnotregularcategory but the category is obtained from the given entry. This command can't be used to test an attribute associated with a multi-entry category.

\glsifcategoryattributetrue{⟨*category*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩}

Expands to ⟨*true*⟩ if the attribute is true and ⟨*false*⟩ otherwise. Expands to ⟨*false*⟩ if there's no such attribute for the given category.

\glsifattributetrue{⟨*entry-label*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩}

As \glsifcategoryattributetrue but the category is obtained from the given entry. Expands to ⟨*false*⟩ if the entry isn't defined. This command can't be used to test an attribute associated with a multi-entry category.

\glsifcategoryattributehasitem{⟨*category*⟩}{⟨*attribute*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

Does ⟨*true*⟩ if the category has the attribute (whose value is a comma-separated list) contains the given item and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category. The item and list are expanded and passed to datatool's \DTLifinlist to perform the test.

11. bib2gls: Managing Glossary Entry Databases

The command line application bib2gls performs two functions in one:

- selects entries from one or more bib files according to information found in the aux file (similar to BibTeX);
- hierarchically sorts entries and collates location lists (similar to makeindex or xindy).

Instead of storing all your entry definitions in a tex file and loading them using \input or \loadglsentries, the entries can instead be stored in a bib file and bib2gls can selectively write the appropriate commands to a glstex file which is loaded using \GlsXtrLoadResources.

Unlike BibTeX, makeindex and xindy, the file created by bib2gls does not contain the code to typeset the (glossary) list. Instead, it creates a file containing the code that defines the entries, using commands like \newglossaryentry and \newabbreviation. The indexing information (such as the location list) is stored in special fields, and the entries are defined in the appropriate order so the glossary can simply be displayed with \printunsrtglossary or, if no glossary is required, the entries can simply be referenced in the document or displayed as standalone definitions (see §8.5).

This means that you can use a bib reference managing system to maintain the database and it reduces the TeX overhead by only defining the entries that are actually required in the document. If you currently have a tex file that contains hundreds of definitions, but you only use a dozen or so in your document, then the build time is needlessly slowed by the unrequired definitions that occur when the file is input. (You can convert an existing tex file containing glossary definitions to a bib file using convertgls2bib, supplied with bib2gls.)

There are some new commands and options added to glossaries-extra to help assist the integration of bib2gls into the document build process. This chapter describes these commands, but it only provides a general overview of bib2gls. The full details and some sample documents are provided in the bib2gls manual.[1] You may prefer to start with the introductory guide:

[1] mirrors.ctan.org/support/bib2gls/bib2gls.pdf

```
texdoc bib2gls-begin
```

There are also some examples in this chapter. See §11.1 for sample bib files.

As with BibTeX and the other indexing applications, bib2gls is a command line application. See "Incorporating makeglossaries or makeglossaries-lite or bib2gls into the document build[2]" for advice on adding bib2gls to your document build. To check that you have bib2gls installed correctly enter the following into your command prompt or terminal, which should show the current version:

```
bib2gls --version
```

Or for a summary of available switches:

```
bib2gls --help
```

Global bib2gls settings are typically set via the command line using switches such as -g or --group. This is different to resources options, such as group, which are only applicable to the given resource set. Since it can be tricky to add command line arguments within some automated build environments, there is now a preamble-only command that may be used to supply those options:

```
\BibGlsOptions{⟨options⟩}
```

The argument is a ⟨*key*⟩=⟨*value*⟩ list of options corresponding to the applicable long switch without the leading -- (such as group for --group but not the short form g) where any --⟨*switch*⟩ that has a corresponding --no-⟨*switch*⟩ becomes a boolean option. For example:

```
\BibGlsOptions{group,collapse-same-location-range=
false}
```

is equivalent to:

```
bib2gls --group --no-collapse-same-location-range
⟨filename⟩
```

This will require at least version 4.0 of bib2gls. From the point of view of glossaries-extra, the \BibGlsOptions command simply adds the information to the aux file, which is

[2] dickimaw-books.com/latex/buildglossaries

picked up by bib2gls. Multiple instances are concatenated and apply to all resource sets. See the bib2gls manual for supported options and further information.

Any options that are referenced by bib2gls before the log file or aux file is read must be set via the command line. This includes localisation support, and the log and aux file input encoding and path.

A document may have one or more *resource sets*. Each resource set corresponds to a glstex file that's created by bib2gls. This file contains the LaTeX code used to define the glossary entries. The trick with this method is that bib2gls writes the code so that the glossary entry definitions match the requested order. This means that \printunsrtglossary can simply be used to list the entries.

To ensure that bib2gls can find out which entries have been used in the document, and their associated indexing information, you need the record package option:

```
\usepackage[record]{glossaries-extra}
```

If you are using hyperref, you may prefer to use record=nameref.

The glstex file created by bib2gls is loaded using:

```
\GlsXtrLoadResources[⟨options⟩]
```

This both sets up the options for the resource set (which bib2gls can detect from the aux file) and inputs the file ⟨*basename*⟩.glstex created by bib2gls (if it exists), where the ⟨*basename*⟩ is \jobname in the first instance and \jobname-⟨*n*⟩ on subsequent instances (where ⟨*n*⟩ is incremented at the end of every \GlsXtrLoadResources). If the optional argument is missing, bib2gls will assume the option src=⟨*basename*⟩. In general, you will need to supply src, particularly if you have multiple resource sets.

\GlsXtrLoadResources will redefine \glsindexingsetting to bib2gls (or bib2gls-xindy or bib2gls-makeindex if record=hybrid). If the upmendex package option was used, then makeindex will be replaced with upmendex.

There is a shortcut command:

```
\glsbibdata[⟨options⟩]{⟨bib-list⟩}
```

This simply does:

```
\GlsXtrLoadResources[src={⟨bib-list⟩},⟨options⟩]
```

If required, the value of ⟨*n*⟩ is stored in the count register:

```
\glsxtrresourcecount
```

although there should be little need to use this.

Note that `\glsxtrresourcefile` is now deprecated. Use `\glsbibdata` instead. (Remember that if you want to share a resource set, including locations, across multiple documents, you need to use the `master` resource option.)

The `\GlsXtrLoadResources` command writes the following to the `aux` file:

```
\glsxtr@resource{⟨options⟩}{⟨basename⟩}
```

This command is searched for by `bib2gls`. If you are using or developing a build system that needs to know which applications to run as part of the document build, you can search the `aux` for instances of `\glsxtr@resource`. For example, using `arara`:

```
% arara: bib2gls if found("aux", "glsxtr@resource")
```

See also Decyphering the Aux File Commands Provided by glossaries.sty and glossaries-extra.sty.[3]

Since the `glstex` file won't exist on the first LaTeX run, the `record` package option additionally switches on `undefaction=warn`. Any use of commands like `\gls` or `\glstext` will produce `??` in the document, since the entries are undefined at this point. Once `bib2gls` has created the `glstex` file the references should be resolved. This may cause a shift in the locations if the actual text produced once the entry is defined is significantly larger than the placeholder `??` (as this can alter the page breaking).

Note that as from v1.12, `\GlsXtrLoadResources` temporarily switches the category code of `@` to 11 (letter) while it reads the file to allow for any internal commands. Be aware of this if you have any entry fields that include the `@` character.

The package options `record=only` and `record=nameref` automatically load glossaries-extra-bib2gls, which provides additional commands that are useful with `bib2gls`. Since they cover sorting and location lists, they're not relevant with the

[3] dickimaw-books.com/latex/auxglossaries/

record=hybrid option.

The information provided in the optional argument of \GlsXtrLoadResources (and therefore also with \glsbibdata) is written to the aux file using:

```
\protected@write\@auxout{\glsxtrresourceinit}
{⟨information⟩}
```

where ⟨*information*⟩ is the information to pass to bib2gls (\glsxtr@resource). The command in the second argument:

```
\glsxtrresourceinit
```

may be used to temporarily redefine commands before the information is written to the file. This does nothing by default, but may be redefined to allow the use of short commands for convenience. For example, with:

```
\renewcommand{\glsxtrresourceinit}{\let\u\glshex}
```

you can just use, for example, \u E6 instead of \string\uE6 in the custom rule. This redefinition of \u is scoped so its original definition is restored after the write operation.

If you have regular expressions or use complex string concatenations with conditionals, such as with assign-fields, then you may find it more convenient to redefine \glsxtrresourceinit to use \GlsXtrResourceInitEscSequences (see §11.6.2).

```
\glsxtrMFUsave
```

If you have mfirstuc v2.08+, \glsxtrMFUsave will be used on the first instance of \GlsXtrLoadResources, and will add \MFUsave to the begin document hook and then disable itself. This is provided to help bib2gls v3.0+ pick up any of mfirstuc's exclusions, blockers and mappings to assist with its sentence case function. The assumption is that all exclusions, blockers and mappings will be set up in the preamble. If there are any within the document environment that you want bib2gls to be aware of, redefine this command to do nothing before the first instance of \GlsXtrLoadResources (or \glsbibdata) and use \MFUsaveatend instead.

If the expansion text is non-empty for the command:

```
\GlsXtrDefaultResourceOptions
```

then this command will be inserted at the start of the resource options. This means that if you have multiple resource commands and you want a default set of options, you can redefine this command or append to it. For example:

```
\appto\GlsXtrDefaultResourceOptions{%
 ignored-type=ignored,
 copy-to-glossary="index" [ type <> "ignored" ]}
```

This should be done before the resource commands to which the options should apply.

11.1. The bib File

The bib file format for bib2gls follows the same syntax as for BibTeX but has different entry types. As with BibTeX, the entry type starts with @ and is case-insensitive. The most common types are: @entry (for general entries), @index (for index entries, typically without a description), @abbreviation (for abbreviations) and @symbol (for symbols). For the complete list of entry types, see the bib2gls user guide.

There are a number of examples in this chapter that use the following sample bib files. The first, terms.bib, provides general entries, which are defined with @entry, and one index-only (no description) entry, which is defined with @index:

```
@entry{bird,
  name={bird},
  description={feathered animal},
  seealso={duck,goose}
}

@entry{duck,
  name={duck},
  description={a waterbird with short legs}
}

@entry{goose,
  name = "goose",
  plural = "geese",
  description={a waterbird with a long neck}
}

@index{example}
```

Entries identified with @entry that are selected by bib2gls will have their data written to

the glstex file in a manner that will define them with \longnewglossaryentry*. Note that this is the starred version which doesn't trim leading and trailing spaces for the description. This is because bib2gls has its own field trimming options. For example, if the "duck" entry is selected, the definition written to the glstex file will effectively be:

```
\longnewglossaryentry*{bird}% label
 {name={bird},seealso={duck,goose}}% fields
 {feathered animal}% description
```

(Although it may additionally have other fields, such as location and group, set.)

The general purpose @entry entry type has the same mandatory fields as \newglossaryentry. That is, the description field and either the name or parent field are required.

Entries identified with @index that are selected by bib2gls will have their data written to the glstex file in a manner that will define them with \newglossaryentry since they are not expected to have a description (although it is permitted, if desired). There are no required fields. If the name field is omitted, the label will be used as the name (unless the parent field has been set). If the description field isn't provided, it will be set to empty. Additionally, the category will be set to index, unless otherwise overridden. So if the "example" entry is selected, the definition written to the glstex file will effectively be:

```
\newglossaryentry{example}{name={example},
 category={index}, description={}}
```

(Again it may additionally have other fields, such as location and group, set.)

The file abbrvs.bib provides some abbreviations (including the awkward nested abbreviation "SHTML" from §5.4) which are defined with @abbreviation:

```
@string{ssi={server-side includes}}
@string{html={hypertext markup language}}

@abbreviation{shtml,
  short="shtml",
  fulllong = ssi # " enabled " # html,
  nestedlong = "\glsps{ssi} enabled \glsps{html}",
  description={a combination of \gls{html} and
\gls{ssi}}
}

@abbreviation{html,
  short ="html",
  long = html,
```

```
  description={a markup language for creating web
pages}
}

@abbreviation{ssi,
  short="ssi",
  long = ssi,
  description={a simple interpreted server-side
scripting language}
}

@abbreviation{xml,
  short={xml},
  long={extensible markup language},
  description={a simple text-base format for
representing structured information}
}

@abbreviation{mathml,
  short={m\BibGlsNoCaseChange{ath}ml},
  long={mathematical markup language},
  description={an \gls{xml}-based language for
describing mathematical notation}
}
```

Note that the awkward `shtml` entry doesn't actually have the `long` field set. Instead I've provided two alternatives in custom fields, `fulllong` and `nestedlong`. In my resource options, I can choose which field to use by aliasing or assigning the applicable custom field to `long`. If I forget to do this, `bib2gls` will warn me that the required `long` field is missing.

The `short` field just has lowercase abbreviations, which is convenient if a smallcaps abbreviation style is chosen. The `short-case-change` resource option can be used to change the case of the `short` field if uppercase is preferred. The MathML abbreviation is awkward as it has a mixed case. The case change can be prevented with `\NoCaseChange` (which `bib2gls` recognises), but this will also prevent any case change with commands like `\GLS`. With `bib2gls` v4.1+, an alternative is to use `\BibGlsNoCaseChange` which only prevents case-changing within `bib2gls` and not in the LaTeX document.

Entries identified with `@abbreviation` that are selected by `bib2gls` will have their data written to the `glstex` file in a manner that will define them with `\newabbreviation`. This means the abbreviation style must be set before `\GlsXtrLoadResources`. Since the style isn't known to `bib2gls`, it assumes that, if the `sort` field is missing (which is best), then the `short` field value should be used as the fallback. If you are using a style that starts the name with the long value, then this fallback will need to be changed (as in examples 156 & 158).

The above abbrvs.bib file defines bib strings (with @string) and uses string concatenation (with #), which is a BibTeX feature. Another supported bib feature is @preamble, which may be used to provide command definitions.

The abbrvs.bib file assumes an abbreviation style where the description must be supplied (such as the long-short-sc-desc style). If this isn't appropriate, bib2gls can be instructed to ignore the description field with the ignore-fields resource option. In this case, the custom nestedlong field may be the more suitable of the two custom fields for the long value (see Example 152).

The dependent ssi and html entries can be detected by bib2gls as it parses the field values so they can automatically be selected even if they aren't recorded. Note, however, that ignored fields don't have their values parsed.

The file symbols.bib provides some symbols which are defined with @symbol:

```
@preamble{"\providecommand{\mtx}[1]{\boldsymbol{#1}}
"}

@symbol{M,
  name={\ensuremath{\mtx{M}}},
  description={a matrix}
}

@symbol{v,
  name={\ensuremath{\vec{v}}},
  description={a vector}
}

@symbol{S,
  name={\ensuremath{\mathcal{S}}},
  description={a set}
}
```

Entries identified with @symbol that are selected by bib2gls will have their data written to the glstex file in a manner that will define them with \longnewglossaryentry* with category={symbol}. This means that if you don't explicitly set the category (either in the bib file or using applicable resource options), then these entries will have the symbol category. You will need to take this into account if you want to define or adjust any hooks that depend on the category.

Example 152 adapts the earlier Example 96 for use with bib2gls but it also references the additional mathml entry. Remember to use the record or record=nameref package option. The latter is better with hyperref and is used for this example: 152

```
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record=nameref]{glossaries-extra}

\setabbreviationstyle{long-em-short-em}

\GlsXtrLoadResources[
 src={abbrvs},
 field-aliases={nestedlong=long},
 ignore-fields={description},
 short-case-change=uc
]
\begin{document}
\tableofcontents

\section{\Glsfmtlong{shtml}}
First use: \gls{shtml}, \gls{html}, \gls{ssi},
\gls{mathml}.

Next use: \gls{shtml}, \gls{html}, \gls{ssi},
\gls{mathml}.
All caps: \GLS{mathml}.

\printunsrtglossaries
\end{document}
```

As with Example 96, the nested \emph from the long-em-short-em abbreviation style, causes the nested HTML and SSI to be typeset in the upright not italic font shape.

Example 152: Using bib2gls: abbreviations

Contents

1 SSI *enabled* HTML

First use: SSI *enabled* HTML (*SHTML*), *hypertext markup language* (*HTML*), *server-side includes* (*SSI*), *mathematical markup language* (*MathML*).
Next use: *SHTML*, *HTML*, *SSI*, *MathML*. All caps: *MATHML*.

Glossary

HTML *hypertext markup language* 1

MathML *mathematical markup language* 1

SHTML SSI *enabled* HTML 1

SSI *server-side includes* 1

Note that this uses resource options to change the definitions *without modifying* the bib files. In more detail, these options are:

```
field-aliases={nestedlong=long},
```

Converts the custom nestedlong field into the long field.

```
ignore-fields={description},
```

Instructs bib2gls to ignore the provided description fields, since the chosen abbreviation style (long-em-short-em) sets the description to the long form (and explicitly setting that field would disrupt the style). Note that if this ignore-fields setting is changed to:

```
omit-fields={"description"},
```

then the xml entry will also be selected (and will therefore appear in the glossary). This is because the reference to that entry can be picked up from the description field with omit-fields but can't with ignore-fields (see Example 154).

The value of the short form is converted to uppercase with:

```
short-case-change=uc
```

Note that you will need bib2gls v4.1+ to provide support for \BibGlsNoCaseChange. For older versions, replace \BibGlsNoCaseChange with \NoCaseChange. (This will alter the behaviour of the all caps \GLS.) If you try this example with bib2gls v4.1+, observe the difference between short-case-change=uc, which uses bib2gls to perform the case-change, and short-case-change=uc-cs, which simply encapsulates the original value with \glsuppercase (which doesn't recognise \BibGlsNoCaseChange as an exclusion).

This means that the definitions that bib2gls writes to the glstex file will be equivalent to those in the preamble of Example 96 (with the additional "MathML" abbreviation).

Example 153 adapts Example 152 to use a small caps abbreviation style. This means that the short-case-change option isn't required. The mixed case in MathML is dealt with by defining \BibGlsNoCaseChange to reduce the font size. An alternative is to use \glstextup instead of \textsmaller, which will match the abbreviation plural suffix. 153

```
\setabbreviationstyle{long-short-sc}
\newcommand{\BibGlsNoCaseChange}[1]{% bib2gls v4.1+
 \textsmaller{#1}}

\GlsXtrLoadResources[
 src={abbrvs},
 field-aliases={nestedlong=long},
 ignore-fields={description}
]
```

This will need the relsize package. Otherwise, the rest of the document is the same as Example 152. However, with the same document build as before, this would lead to the odd-looking capital "S" at the start of the section header (Ssi enabled HTML) which is caused by \Glsfmtlong trying to apply sentence-casing to \glsps: the \makefirstuc mapping will replace \glsps with \Glsps. This wasn't noticeable with the previous example which had uppercase abbreviations. It is noticeable with small caps.

The trick to prevent this is to insert an empty group before the leading \glsps. Rather than edit the abbrvs.bib file, bib2gls can be instructed to implement this fix by invoking bib2gls with the --mfirstuc-protection switch followed by the list of fields that need to be adjusted. In this case, only the long field needs to be checked:

```
bib2gls --mfirstuc-protection long myDoc
```

Or use \BibGlsOptions:

```
\BibGlsOptions{mfirstuc-protection={long}}
```

An inspection of the resulting glstex file will show that only the long field for the shtml entry has been adjusted, as it's the only one that starts with a known problematic command.

Example 153: Using bib2gls: smallcap abbreviations

Contents

1 SSI enabled HTML

First use: SSI enabled HTML (SHTML), hypertext markup language (HTML), server-side includes (SSI), mathematical markup language (MATHML).

Next use: SHTML, HTML, SSI, MATHML. All caps: MATHML.

Glossary

HTML hypertext markup language 1

MATHML mathematical markup language 1

SHTML SSI enabled HTML 1

SSI server-side includes 1

Example 154 adapts Example 153 to use omit-fields instead of ignore-fields and uses the custom fulllong for the long value. The use of omit-fields rather than ignore-fields means that bib2gls still parses the original description provided in the bib file, even though that field is omitted from the glstex file. This means that the xml entry is selected even though it hasn't been referenced anywhere in the document, but I've used gather-parsed-dependencies to gather the dependencies found in the description field and add them to the seealso field so that there is now a cross-reference to them in the glossary. Note that this requires bib2gls v4.1+. 154

```
\GlsXtrLoadResources[
 src={abbrvs},
 field-aliases={fulllong=long},
 omit-fields={"description"},
 gather-parsed-dependencies
]
```

The `omit-fields` option has a complex string concatenation syntax with optional conditionals. This example doesn't have a conditional part but it's important to quote the string part (see the `bib2gls` user guide for more details). The rest of the document is as for Example 153, but note that now the `--mfirstuc-protection` option should not be used as the `long` field is no longer problematic.

Example 154: Using bib2gls: gathering dependencies from field values

Contents

1 Server-side includes enabled hypertext markup language

First use: server-side includes enabled hypertext markup language (SHTML), hypertext markup language (HTML), server-side includes (SSI), mathematical markup language (MATHML).

Next use: SHTML, HTML, SSI, MATHML. All caps: MATHML.

Glossary

HTML hypertext markup language 1

MATHML mathematical markup language 1, *see also* XML

SHTML server-side includes enabled hypertext markup language 1, *see also* HTML & SSI

SSI server-side includes 1

XML extensible markup language

11.2. Indexing (Recording)

As with makeindex and xindy, the \gls-like and \glstext-like commands automatically index, but the underlying indexing mechanism is more like that used with \makenoidxglossaries. Each indexing instance creates a record in the aux file, which bib2gls can then pick up when it parses the aux file. Each record has a location, an associated format (the location encap) which can be set with the format key, and an associated location counter (as with the other indexing methods).

The formatted location list is stored in the location field (unless save-locations=false). Additionally, the individual locations are stored in the loclist field as an etoolbox internal list (as with \makenoidxglossaries). This may be used to pick out individual

locations to avoid the complexity of parsing the formatted list.

See the "Location List Options" section of the bib2gls manual for information on how to adjust the way the location list is compacted or how to separate the location list into groups associated with different counters or how to move locations with a particular format into a separate field.

11.3. Selection

Commands like \glsaddall and \glsaddallunused don't work with bib2gls as these commands have to iterate over each glossary's internal lists of defined entry labels, which will be empty on the first run and on subsequent runs will only contain those entries that have been selected by bib2gls. Use selection=all to select all entries instead.

The default behaviour is for bib2gls to select all entries that have a record in the aux file, and any dependent entries (including parent and cross-references). This behaviour can be changed with the selection resource option, and there are other options that can apply a filter to adjust the selection (see the bib2gls manual for further details).

The glsignore format (for example, \gls[format=glsignore]{duck}) is recognised by bib2gls as a special ignored record. This means that it will match the "recorded" selection criteria, but the record won't be added to the location list, so you won't get spurious commas in the location list caused by blank locations (as can happen with the other indexing methods).

If an entry has one or more ignored record, no other type of record, and no dependencies then it is considered an "ignored entry". By default it behaves in the same way as all other recorded entries, but ignored entries can be tidied away into an ignored glossary with the ignored-type resource option. This ensures that the entries are defined but they won't appear in the regular glossaries.

For example, at the start of the front matter, set the default to the ignored format:

```
\GlsXtrSetDefaultNumberFormat{glsignore}
```

then at the start of the main matter set the default to the normal format:

```
\GlsXtrSetDefaultNumberFormat{glsnumberformat}
```

This will prevent any records in the front matter from occurring in the location lists. If you have any entries that only occur in the front matter and not the main matter then they will end up in the glossary with no location list. If this is undesirable, move them to an ignored glossary:

```
\GlsXtrLoadResources[ignored-type=ignored]
```

(This option requires bib2gls v4.0+.) This will automatically define an ignored glossary labelled "ignored" with the unstarred \provideignoredglossary.

The selection option indicates which entries should be selected from the bib files (listed in src). For example, selection=all indicates to select all entries, regardless of whether or not the entries have been referenced in the document. This may lead to empty location lists for some (or all) entries. The default setting is selection=recorded and deps, which indicates to select all entries that have records and any dependent entries. See the bib2gls user manual for more details of this option.

11.4. Sorting and Displaying the Glossary

With makeindex and xindy, the indexing data (read from the associated input file created by \makeglossaries) is sorted and the code to typeset the glossary is written to an output file, which is then input by \printglossary. These general purpose indexing applications only have access to the indexing data, which consists of the sort value, the "actual text" (\glossentry{⟨*label*⟩} or \subglossentry{⟨*level*⟩}{⟨*label*⟩}), and the locations or see/seealso/alias information. This means that the indexing applications can't detect any references within fields such as description, which means those references won't be available to the indexing applications until the applicable field has been typeset in the document (typically, when the description is typeset in the glossary).

With bib2gls, the entries supplied in the bib files are selected according to the designated criteria and sorted. Then the entry definition code (\longnewglossaryentry or \newabbreviation or \newglossaryentry) is written to the glstex file in the order obtained by sorting. This means that the glossary's internal list only contains the required entry labels, and those labels are in the required order, so the glossary can be displayed with \printunsrtglossary (see §8.4). Since bib2gls has access to all fields, it's able to detect dependent entries, even if they haven't been recorded. This means that those dependent entries can still be selected, but they won't have any records until they are actually indexed. This means that the entries will be defined in the glstex file but may not have a location list until the next LaTeX+bib2gls+LaTeX run.

The source data from bib2gls's point of view is in bib files using the bib format described in bib2gls's user guide. The LaTeX entry definitions (using commands provided by glossaries and glossaries-extra) are written to the glstex file that's automatically

input by the corresponding \GlsXtrLoadResources.

With bib2gls, the processed information, such as the location list or letter group, is stored in fields such as location or group (where applicable), so the information can be included by \printunsrtglossary, but it means that the information is also available for use elsewhere in the document (so the savenumberlist package option provided by glossaries is redundant). This is different from indexing applications where the processed information is embedded in the typesetting code, which makes it hard to extract.

There are many sorting options provided by bib2gls. The default is sort=resource, which means to sort alphabetically according to the resource set's locale. This will typically match the document's language setting, if a single language is set up, or Java's default locale if no language support was provided. (The language tag obtained from tracklang's interface is written to the aux file but, for multiple languages, there's no way for bib2gls to automatically detect which language applies to which resource set.)

For a multilingual document you need to explicitly set the locale using a well-formed language tag. This can be done with the locale option, which not only affects sorting, but also other language-sensitive features, such as numeric and date/time parsing, or it can be done via the sort option if the locale is only applicable to language-sensitive sorting. For example:

```
\newglossary*{german}{\glossaryname}
\newglossary*{english}{\glossaryname}
\GlsXtrLoadResources[
  type=german, % glossary for these entries
  src={terms-de}, % data in terms-de.bib
  sort=de-DE-1996 % sort according to this locale
]
\GlsXtrLoadResources[
  type=english, % glossary for these entries
  src={terms-en}, % data in terms-en.bib
  sort=en-GB % sort according to this locale
]
```

The locale-sensitive sort methods usually ignore most punctuation, so for lists of symbols you may find it more appropriate to use one of the letter-base sort methods that sort according to the Unicode value of each character or you can sort by a different field, such as the description or the entry label. Alternatively you can provide a custom rule. See the bib2gls manual for full details of all the available sort methods.

Suppose the bib examples shown earlier have been stored in the files terms.bib (general entries), abbrvs.bib (abbreviations) and symbols.bib (symbols) which may either be in the current directory or on TEX's path. Example 155 requires these three files but has a single resource command. Remember that the abbreviation style must be set before \GlsXtrLoadResources. Also, the abbrvs.bib file contains the awkward nested abbreviation that

155

doesn't have the long field set, so I need to choose one of my custom fields to use for the long form. In this case, I've chosen fulllong.

```
\documentclass{article}

\usepackage[record]{glossaries-extra}

\setabbreviationstyle{long-short-sc-desc}

\GlsXtrLoadResources[src={terms,abbrvs,symbols},
 field-aliases={fulllong=long},
 description-case-change=firstuc,
 post-description-dot=all
]

\begin{document}
\gls{bird}

\gls{shtml}

\gls{M}

\GlsXtrSetDefaultNumberFormat{glsignore}
\printunsrtglossaries
\end{document}
```

The document build process (assuming the document is called mydoc) is:

```
pdflatex mydoc
bib2gls mydoc
pdflatex mydoc
```

Example 155: Using bib2gls: one resource set

bird
server-side includes enabled hypertext markup language (SHTML)
$\boldsymbol{M}$

Glossary

bird Feathered animal. 1, *see also* duck & goose

duck A waterbird with short legs.

goose A waterbird with a long neck.

hypertext markup language (HTML) A markup language for creating web pages.

$\boldsymbol{M}$ A matrix. 1

server-side includes enabled hypertext markup language (SHTML) A combination of hypertext markup language (HTML) and server-side includes (SSI). 1

server-side includes (SSI) A simple interpreted server-side scripting language.

This creates a single glossary containing the entries: bird, duck, goose, html, M, shtml and ssi (in that order). The bird, shtml and M entries were added because bib2gls detected (from the aux file) that they had been used in the document. (That is, they had records.) The other entries were added because bib2gls detected (from the bib files) that they are referenced by the recorded entries. In the case of duck and goose, they are in the seealso field for bird. In the case of ssi and html, they are referenced in the description field of shtml. These cross-referenced entries won't have a location list when the glossary is first displayed, but depending on how they are referenced, they may pick up a location list on the next document build. (This example switches to format=glsignore just before the glossary so it doesn't make a difference.) The xml entry isn't required at all, and so hasn't been defined (from LaTeX's point of view).

Example 156 uses the same three bib files but the entries are separated into different glossaries with different sort methods. However, I would like them all to apply the adjustments to the description field, so I've redefined \GlsXtrDefaultResourceOptions to expand to my preferred default.

156

```
\usepackage[record,abbreviations,symbols]
{glossaries-extra}

\renewcommand\GlsXtrDefaultResourceOptions{
 description-case-change=firstuc,
 post-description-dot=all
}

\setabbreviationstyle{long-short-sc-desc}

\GlsXtrLoadResources
 [src={terms},sort=en-GB,type=main]

\GlsXtrLoadResources
 [src={abbrvs},sort=letter-nocase,
 abbreviation-sort-fallback={long},
 field-aliases={fulllong=long},
 type=abbreviations]

\GlsXtrLoadResources
 [src={symbols},sort=use,type=symbols]

\begin{document}
\gls{bird}

\gls{shtml}

\gls{M}

\GlsXtrSetDefaultNumberFormat{glsignore}
\printunsrtglossaries
\end{document}
```

This is somewhat contrived as there's little reason to use a non-locale sort method for word abbreviations, although it doesn't make a difference in this particular example. However, it may be appropriate for abbreviations that contain punctuation or symbols that need to be taken into account when sorting.

The default `abbreviation-sort-fallback={short}` will just use the `short` value as the fallback if the `sort` field isn't set. If the abbreviation style, as in this case, puts the long form in the `name` field, then this fallback setting may need adjusting. Remember that by *not* setting the `sort` field the fallback mechanism can be used to make adjustments for individual documents without having to modify the source `bib` files.

Example 156: Using bib2gls: multiple resource sets

bird
server-side includes enabled hypertext markup language (SHTML)
$\boldsymbol{M}$

Glossary

bird Feathered animal. 1, *see also* duck & goose

duck A waterbird with short legs.

goose A waterbird with a long neck.

Symbols

$\boldsymbol{M}$ A matrix. 1

Abbreviations

hypertext markup language (HTML) A markup language for creating web pages.

server-side includes (SSI) A simple interpreted server-side scripting language.

server-side includes enabled hypertext markup language (SHTML) A combination of hypertext markup language (HTML) and server-side includes (SSI). 1

Example 157, on the other hand, has multiple instances of `\GlsXtrLoadResources` with the same `type`, which will produce a glossary with sub-blocks. For example: 157

```
\usepackage[record,style=indexgroup]
 {glossaries-extra}

\renewcommand\GlsXtrDefaultResourceOptions{
 description-case-change=firstuc,
 post-description-dot=all
}
```

```
\setabbreviationstyle{long-short-sc-desc}

\GlsXtrLoadResources
 [src={abbrvs},sort=letter-nocase,type=main,
  field-aliases={fulllong=long},
  group=abbreviations]
\glsxtrsetgrouptitle{abbreviations}{Abbreviations}

\GlsXtrLoadResources
 [src={symbols},sort=use,type=main,
  group=symbols]
\glsxtrsetgrouptitle{symbols}{Symbols}

\GlsXtrLoadResources[src={terms},sort=en-GB,type=main]

\begin{document}
\gls{bird}

\gls{shtml}

\gls{M}

\GlsXtrSetDefaultNumberFormat{glsignore}
\printunsrtglossaries
\end{document}
```

This sets the `group` field for the first two resource sets to the label given by the `group` resource option. In general, the `group` resource option should only be used when splitting up a single glossary into sub-groups in this way. Normally, the group should be automatically set as a by-product of the sorting process (if enabled).

If multiple instances of `\GlsXtrLoadResources` assign entries to the same glossary, the glossary will end up with sub-blocks (that may or may not contain one or more groups), where each sub-block is in the order of the `\GlsXtrLoadResources` commands, but the entries within each sub-block are in the designated sort order of the corresponding `\GlsXtrLoadResources` command. There won't necessarily be any visual difference between the sub-blocks.

Example 157 therefore results in a single glossary composed of three sub-blocks. The first sub-block has a single group with the label `abbreviations` and title "Abbreviations"; the second sub-block again has a single group, this time with the label `symbols` and title "Symbols"; the third sub-block consists of the usual letter groups.

Note that for this example to work, you must run `bib2gls` with the `--group` (or `-g`) switch. For example, if the document is called `myDoc.tex`:

```
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
```

This example is somewhat contrived as it isn't usual to split up a glossary that only has alphabetic entries in this way. It would more typically be used to separate a punctuation group (which needs to be ordered by character code) from the normal alphabetical groups (as is done for the Index of this user manual).

Example 157: Using bib2gls: sub-blocks

bird
server-side includes enabled hypertext markup language (SHTML)
$\boldsymbol{M}$

Glossary

Abbreviations

hypertext markup language (HTML) A markup language for creating web pages.

server-side includes enabled hypertext markup language (SHTML) A combination of hypertext markup language (HTML) and server-side includes (SSI). 1

server-side includes (SSI) A simple interpreted server-side scripting language.

Symbols

$\boldsymbol{M}$ A matrix. 1

B

bird Feathered animal. 1, *see also* duck & goose

D

duck A waterbird with short legs.

G

goose A waterbird with a long neck.

The value of the group field must always be a label (so avoid special characters). You can set the corresponding title with glsxtrsetgrouptitle (see §8.6). If no title is set then the label is used as the group title. The group field does not influencing sorting (it's usually set as a by-product of sorting). Setting this field overrides normal behaviour and can cause fragmented groups (as in Example 139).

Example 158 adapts Example 155 so that it still has only one resource set but now has four glossaries: "main" (the default), "abbreviations", "symbols", and "index". 158

Entries defined with @abbreviation in the bib file are written to the glstex file in such a way that they will be defined with \newabbreviation when the glstex file is input by \GlsXtrLoadResources, which means they will automatically be added to the "abbreviations" glossary, if the type field isn't set.

Entries defined with @symbol in the bib file are written to the glstex file in such a way that they will be defined with \longnewglossaryentry* which means they will automatically be added to the default "main" glossary, if the type field isn't set. Similarly for @index, which will be defined in the glstex file with \newglossaryentry.

Rather than edit the bib files to set the type field explicitly, which would make the bib files less adaptable for other documents, the assign-fields option is used to set the type field for this particular resource set. In this example, type is set to "symbols" for any entries that are defined with @symbols and to "index" for any entries that are defined with @index. The remaining entries will be added to the default main glossary.

This means that even though only one resource set is used, it's possible to distribute the entries into different glossaries, but they will all be sorted together. Since the sort field hasn't been set for any of the entries, the appropriate fallback fields will be used. These are changed for the @symbol and @abbreviation entries.

The sort fallback fields are only used when the sort field *isn't* set (bib2gls *falls back on* the applicable fallback field to compensate for the missing sort field). If the sort field is set, then no fallback is required. This fallback system allows for a more flexible approach to sorting.

The final glossary "index" has a copy of all selected entries. This is done with the copy-to-glossary resource option. Since all the entries were sorted together, they will be copied to the "index" glossary in their sorted order.

This duplication can cause a problem with duplicate hypertargets if hyperref is loaded. This problem is avoided by redefining \glstarget to \glsxtrtarget in the preamble (before any targets are created).

Unlike the previous examples, this example also uses the "example" entry, which is defined in the terms.bib file with @index. This should only go in the index not in the main glossary (which is achieved with the assign-fields option), but it shouldn't be duplicated by the copy-to-glossary option (which is ensured by the definition of \bibglscopyto-

glossary provided in the glstex file).

```
\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record,
 nostyles,stylemods={topic,tree,bookindex},
 style=tree,abbreviations,symbols,index]
 {glossaries-extra}

\renewcommand{\glstarget}{\glsxtrtarget}

\setabbreviationstyle{long-short-sc-desc}

\GlsXtrLoadResources[src={terms,abbrvs,symbols},
 assign-fields={
    type = "symbols"
       [ entrytype->actual = "symbol" ],
    type = "index" [ entrytype->actual = "index" ]
  },
 sort=en-GB,
 symbol-sort-fallback=name,
 abbreviation-sort-fallback=long,
 field-aliases={fulllong=long},
 copy-to-glossary={"index"},
 description-case-change=firstuc,
 post-description-dot=all
]

\newcommand{\primaryfmt}[1]{\hyperbf{#1}}

\begin{document}
\gls{bird} \gls{example}.

\gls{shtml}

\gls{M}

\newpage

\renewcommand*{\glsextrapostnamehook}[1]{%
 \glsadd[format=primaryfmt]{#1}}%
\printunsrtglossary*[nonumberlist]
```

```
 {\glssetcategoryattribute{general}{glossname}{firstuc}}
\printunsrtabbreviations
 [nonumberlist,style=topic]
\printunsrtsymbols[nonumberlist]

\renewcommand*{\glsextrapostnamehook}[1]{}%
\printunsrtindex[style=bookindex]
\end{document}
```

This uses several different approaches to changing the case of field values. The `description-case-change` resource option instructs `bib2gls` to alter the value of the `description` field to ensure that it starts with an uppercase letter. However, the case change for the `name` field is performed by the topic style, for the list of abbreviations, and by locally setting the `glossname` attribute for the `general` category for the main glossary. The case change for the `name` field could also be performed by `bib2gls`, but that would affect the index as well.

The general purpose post-name hook `\glsextrapostnamehook` is used to automatically index each entry's appearance in their own list using a custom format. Indexing within the glossary can only occur once the glossary is typeset, which can't happen until the entries have been defined. This means that an extra `bib2gls` run is needed.

```
pdflatex mydoc
bib2gls mydoc
pdflatex mydoc
bib2gls mydoc
pdflatex mydoc
```

The second page for Example 158 is shown on page 566.

This document again only explicitly references the "bird", "html" and "M" entry. The other entries that appear in the final document were selected because they were dependents of the selected entries. These dependent entries end up being indexed in the second LaTeX run (following the first `bib2gls` run) because they are now in the glossary and so are automatically indexed. They are then selected on the next `bib2gls` because they now have records in the `bib` file (in addition to being flagged as dependencies).

This means that if the `bib` files are later edited to remove the dependencies, the formerly dependent entries will continue to be selected (because of the `\glsadd` in the post-name hook) until the associated `glstex` file is deleted.

You can provide your own custom sort rule. For example, if you are using XeLaTeX or LuaLaTeX or a recent LaTeX kernel:

Example 158: Using bib2gls: one resource set but four lists

Glossary

Bird Feathered animal.
Duck A waterbird with short legs.
Goose A waterbird with a long neck.

Abbreviations

Hypertext markup language (HTML)

A markup language for creating web pages.

Server-side includes (SSI)

A simple interpreted server-side scripting language.

Server-side includes enabled hypertext markup language (SHTML)

A combination of hypertext markup language (HTML) and server-side includes (SSI).

Symbols

$\boldsymbol{M}$ A matrix.

Index

```
\GlsXtrLoadResources[
  src={terms}, % entries in terms.bib
  sort=custom, % custom sort rule
  sort-rule={% required with sort=custom
  < æ;Æ < a;á;å;ä,Ä;Á;Å;Ä < b,B
  < c;ć,C;Ć < d,D < e;é,E;É < f,F < g,G
  < h,H < i;í,I;Í < j,J < l;ł,L;Ł < m,M < n,N
  < o;ö;ø,O;Ö;Ø < p,P < q,Q < r,R < s;ś,S;Ś
  < t,T < u;ú,U;Ú < v,V < w,W < x,X < y,Y <
  z;ż,Z;Ż
  }
]
```

With old versions of the LaTeX kernel, UTF-8 characters, such as é or ø, will expand when written to the `aux` file.

Some of the options, including `sort-rule`, allow Unicode characters to be indicated in the format `\u`⟨*hex*⟩ (or `\u` ⟨*hex*⟩). `bib2gls` will recognise this as the character given by the hexadecimal value ⟨*hex*⟩.

The ⟨*options*⟩ provided to `\GlsXtrLoadResources` will expand as they are written to the `aux` file (unless protected). This includes `\u`, so with a non-Unicode aware engine or where the document source is required to be ASCII, the character æ needs to be written as `\string\u E6` and so on. Alternatively, use the shortcut `\string\u E6`.

For example, the above can be rewritten as:

```
\GlsXtrLoadResources[
  src={terms}, % entries in terms.bib
  sort=custom, % custom sort rule
  sort-rule={% required with sort=custom
  < \glshex E6;\glshex C6
  < a;\glshex E1;\glshex E5,\glshex E4,A;\gls-
hex C1;\glshex C5;\glshex C4
  < b,B < c;\glshex 0107,C;\glshex 0106 < d,D
  < e;\glshex E9,E;\glshex C9 < f,F < g,G
  < h,H < i;\glshex ED,I;\glshex CD < j,J
```

```
  < l;\glshex 0142,L;\glshex 0141 < m,M < n,N
  < o;\glshex F6;\glshex F8,O;\glshex D6;\glshex D8
  < p,P < q,Q < r,R < s;\glshex 013F,S;\glshex 015A
  < t,T < u;\glshex FA,U;\gls-
hex DA < v,V < w,W < x,X < y,Y
  < z;\glshex 017C,Z;\glshex 017B
  }
]
```

Bear in mind that the rule-based comparators (that is, the localisation rules and the custom rule) break up terms according to `break-at`, which allows punctuation and spaces to be stripped from sort values. The default is `break-at=word`, which replaces content between word boundaries with a marker (identified with `break-marker`) to perform word-sorting. This means that if you actually want the punctuation retained in the sort value, you will need to use `break-at=none` instead.

11.5. Record Counting

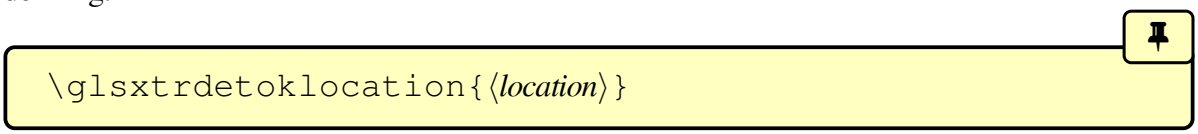
As from version 1.1 of `bib2gls`, you can save the total record count for each entry by invoking `bib2gls` with the `--record-count` or `--record-count-unit` switches. These options will ensure that when each entry is written to the `glstex` file `bib2gls` will additionally set the following internal fields for that entry:

- `recordcount`: set to the total number of records found for the entry;
- `recordcount.`⟨*counter*⟩: set to the total number of records found for the entry for the given counter.

If `--record-count-unit` is used then additionally:

- `recordcount.`⟨*counter*⟩`.`⟨*location*⟩: set to the total number of records found for the entry for the given counter with the given location.

Only use the unit counting option if the locations don't contain any special characters. With hyperref use `\the`⟨*counter-name*⟩ rather than `\the`⟨*counter-name*⟩. Otherwise, if you really need unit counting with locations that may contain formatting commands, then you can try redefining:



```
\glsxtrdetoklocation{⟨location⟩}
```

so that it detokenizes ⟨*location*⟩ but take care when using `\GlsXtrLocationRecordCount` with commands like `\thepage` as they can end up becoming detokenized too early.

Note that the record count includes locations that `bib2gls` discards, such as ignored records, duplicates and partial duplicates (unless you filter them out with `--record-count-rule`).

It doesn't include cross-reference records. For example, suppose a document has an entry with the label `bird` that is recorded (indexed) as follows:

Page 1 two (2) instances of `\gls{bird}`;

Page 2 one (1) instance of `\gls{bird}`;

Page 3 four (4) instances of `\gls{bird}`;

Section 3 one (1) instance of `\gls[counter=section]{bird}`.

Then the total record count (stored in the `recordcount` field) is $2+1+4+1=8$, the total for the page counter (stored in the `recordcount.page` field) is $2+1+4=7$, and the total for the section counter (stored in the `recordcount.section` field) is 1.

With the unit counting on as well, the following fields are assigned:

- `recordcount.page.1` is set to 2;
- `recordcount.page.2` is set to 1;
- `recordcount.page.3` is set to 4;
- `recordcount.section.3` is set to 1.

You can access these fields using the following commands which will expand to the field value if set or to 0 if unset:

```
\GlsXtrTotalRecordCount{⟨entry-label⟩}
```

This expands to the total record count for the entry given by ⟨*label*⟩. For example:

```
\GlsXtrTotalRecordCount{bird}
```

expands to 8.

```
\GlsXtrRecordCount{⟨entry-label⟩}{⟨counter⟩}
```

This expands to the counter total for the entry given by ⟨*entry-label*⟩ where ⟨*counter*⟩ is the counter name. For example:

```
\GlsXtrRecordCount{bird}{page}
```

expands to 7 and

```
\GlsXtrRecordCount{bird}{section}
```

expands to 1.

```
\GlsXtrLocationRecordCount{⟨entry-label⟩}{⟨counter⟩}{⟨location⟩}
```

This expands to the total for the given location. For example

```
\GlsXtrLocationRecordCount{bird}{page}{3}
```

expands to 4. Be careful about using `\thepage` in the ⟨*location*⟩ part. Remember that due to TEX's asynchronous output routine, `\thepage` may not be correct.

There are commands analogous to the entry counting commands like `\cgls` and `\cglsformat` that are triggered by the record count. These are listed below. The test to determine if the entry's record count exceeds the trigger value (which should be stored in the `recordcount` attribute) is obtained with:

```
\glsxtrifrecordtrigger{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
```

If the `recordcount` attribute is set and ⟨*total*⟩ exceeds the value given by the `recordcount` attribute, then this does ⟨*true*⟩ otherwise it does ⟨*false*⟩. The ⟨*total*⟩ is given by:

```
\glsxtrrecordtriggervalue{⟨entry-label⟩}
```

This should expand to the record count value that needs testing. The default definition is:

```
\newcommand*{\glsxtrrecordtriggervalue}[1]{%
 \GlsXtrTotalRecordCount{#1}%
}
```

This command may be redefined as appropriate. For example, it may be redefined to use `\GlsXtrRecordCount` for a particular location counter or to use `\GlsXtrLocationRecordCount` for a particular location.

The `recordcount` attribute may be set with `\glssetcategoryattribute` or can be set for each listed category with:

```
\GlsXtrSetRecordCountAttribute{⟨category-list⟩}{⟨value⟩}
```

The value must be an integer.

Commands like \rgls behave slightly differently to \cgls. It's necessary for the command to add a record to the aux file in order for the entry to be selected and for the record count to be correct on the next bib2gls+LaTeX run (for the default selection={recorded and deps}). The trigger record is created with format=glstriggerrecordformat, which bib2gls v1.1+ recognises as a special type of ignored location format. This corresponds to the command:

\glstriggerrecordformat{⟨*location*⟩}

As with \glsignore, this command does nothing and is considered an ignored record (so it won't appear in the location list), but it indicates to bib2gls that the entry must be selected and, if the trigger-type option has been set, the entry will be assigned to the trigger-type glossary.

For example, to assign the entry to an ignored glossary:

```
\newignoredglossary{ignored}
\GlsXtrLoadResources[trigger-type=ignored]
```

This ensures that the entry is defined but it won't show up the normal glossary.

The post-link hook won't be implemented if the record trigger is tripped. (That is, if the \rglsformat-like command is used instead of the \gls-like command.)

\rgls[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \gls otherwise it creates a trigger record and uses:

\rglsformat{⟨*entry-label*⟩}{⟨*insert*⟩}

This has the same definition as \cglsformat.

\rglspl[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \glspl otherwise it creates a trigger record and uses:

\rglsplformat{⟨entry-label⟩}{⟨insert⟩}

which uses the appropriate plural fields.

\rGls[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \Gls otherwise it creates a trigger record and uses:

\rGlsformat{⟨entry-label⟩}{⟨insert⟩}

which performs the appropriate case-change.

\rGlspl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \Glspl otherwise it creates a trigger record and uses:

\rGlsplformat{⟨entry-label⟩}{⟨insert⟩}

which uses the appropriate plural fields and case-change.

\rGLS[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \GLS otherwise it creates a trigger record and uses:

\rGLSformat{⟨entry-label⟩}{⟨insert⟩}

which performs the appropriate case-change.

\rGLSpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] *modifiers:* * + ⟨*alt-mod*⟩

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \GLSpl otherwise it creates a trigger record and uses:

\rGLSplformat{⟨entry-label⟩}{⟨insert⟩}

which uses the appropriate plural fields and case-change.

To make it easier to switch on record counting for an existing document, you can use:

```
\glsxtrenablerecordcount
```

This redefines \gls, \glspl, \Gls, \Glspl, \GLS, \GLSpl to \rgls, \rglspl, \rGls, \rGlspl, \rGLS, \rGLSpl, respectively, for convenience. This command will also switch the shortcut commands such as \ac or \ab, if they have been enabled, from using the \cgls-like commands to the corresponding \rgls command.

For example, using the earlier terms.bib, abbrvs.bib and symbols.bib example files:

```
\documentclass{article}

\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}

\newignoredglossary{ignored}

\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources[
 src={terms,abbrvs,symbols},
 field-aliases={fulllong=long},
 trigger-type=ignored,
 category={same as entry}
]

\glsxtrenablerecordcount
\GlsXtrSetRecordCountAttribute{general,abbreviation}
{1}

\glsdefpostlink{entry}{\glsxtrpostlinkAddDescOn-
FirstUse}
\glsdefpostlink{symbol}{\glsxtrpostlinkAddDescOn-
FirstUse}

\begin{document}
\gls{bird}, \gls{ssi}, \gls{bird}, \gls{html},
\gls{M}, \gls{html}.

\printunsrtglossaries
```

```
\end{document}
```

If the document is called myDoc.tex, then the build process is:

```
pdflatex myDoc
bib2gls --record-count myDoc
pdflatex myDoc
```

The category={same as entry} resource option assigns the category field to the bib entry type (without the initial @). This means that the entries defined in terms.bib (with @entry) have their category set to entry, the entries defined in abbrvs.bib (with @abbreviation) have their category set to abbreviation, and the entries defined in symbols.bib (with @symbol) have their category set to symbol.

I've added post-link hooks to the entry and symbol categories to show the description on first use (but not for the abbreviation category).

Example 159: Using bib2gls: record counting

M (a matrix), server-side includes, bird (feathered animal), hypertext markup language (HTML), ***M***, HTML.

Glossary

bird feathered animal 1, *see also* duck & goose

duck a waterbird with short legs

goose a waterbird with a long neck

hypertext markup language (HTML) a markup language for creating web pages 1

M a matrix 1

In the above, ssi and bird only have one record. However, they have been treated differently. The ssi entry is using \rglsformat whereas the bird entry is using the normal \gls behaviour. This is because the record counting hasn't been applied to the custom entry category, whereas it has been applied to the abbreviation and symbol categories.

11.5.1. Unit Record Counting

If you want unit record counting you need to remember to invoke bib2gls with --record-count-unit and you will also need to redefine \glsxtrrecordtriggervalue

appropriately. For example, suppose you want to reset all abbreviations at the start of each chapter, so that the full form is shown again, but only if the abbreviation isn't used elsewhere in the chapter.

This would require records with the counter set to chapter. This can be done with the counter package option:

```
\usepackage[record,counter=chapter]{glossaries-
extra}
```

However, this will have chapter numbers instead of page numbers in the location lists. If you don't want location lists then this isn't a problem. The list can simply be suppressed with nonumberlist.

If you want page numbers in the location lists then you will need to record each entry with both the page and chapter counters. This can be done with the hook that occurs before the \gls options are set:

```
\renewcommand{\glslinkpresetkeys}{%
 \glsadd[format=glsignore,counter=chapter]{\gls-
label}}
```

Note that I've used the ignored location format to prevent the chapter number from being added to the location list. An alternative is to use the loc-counters=page resource option to only show the locations that use the page counter.

The definition of \glsxtrrecordtriggervalue needs to be changed so that it uses the total for the given location. If hyperref is used, you will need \theHchapter:

```
\renewcommand*{\glsxtrrecordtriggervalue}[1]{%
 \GlsXtrLocationRecordCount{#1}{chapter}{\theH-
chapter}%
}
```

otherwise use \thechapter.

Consider the following (using the abbreviations defined in the earlier abbrvs.bib):

```
\begin{document}
\chapter{First}
\gls{html}. \gls{html}.  \gls{html}. \gls{ssi}.

\chapter{Second}
```

```
\gls{html}. \gls{ssi}.  \gls{ssi}.
\gls{xml}.

\printunsrtglossaries
\end{document}
```

Note that the `xml` entry is only used once in the entire document, but it will still be added to the glossary.

The previous example used the `trigger-type` resource option to move entries with the `glstriggerrecordformat` encap (that is, they didn't exceed the trigger value) to another glossary. Unfortunately, using that option in this case will move all three abbreviations to the `trigger-type` glossary. The `ssi` entry is only used once in the first chapter (but is used twice in the second chapter), and the `html` is only used once in the second chapter (but is used three times in the first chapter). So all three will have records in the `aux` file with the special `glstriggerrecordformat` format.

A simple solution is to omit any entries that don't have the `location` field set when displaying the glossary:

```
\renewcommand*{\printunsrtglossaryentryprocesshook}
[1]{%
 \glsxtrifhasfield*{location}{#1}
 {}{\printunsrtglossaryskipentry}%
}
\printunsrtglossaries
```

An alternative is to test the total record count, but remember that each entry is being recorded twice: once with the page counter and once with the chapter counter, so the total count for the `ssi` entry will be 2 not 1.

Take care not to strip entries from a hierarchical glossary as it will break the hierarchy and will cause formatting problems in the glossary.

The complete document is:

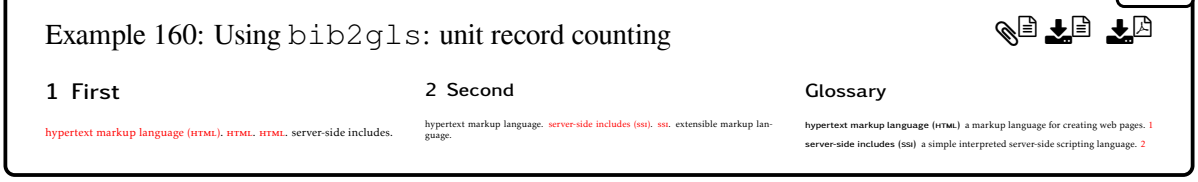
```
\documentclass{scrreport}
\usepackage[T1]{fontenc}
\usepackage{kpfonts}
\usepackage[colorlinks]{hyperref}
\usepackage[record,postdot]{glossaries-extra}
```

```
\setabbreviationstyle{long-short-sc-desc}

\GlsXtrLoadResources[src={abbrvs},
 field-aliases={fulllong=long}
]

\preto{\chapter}{\glsresetall}
\glsxtrenablerecordcount
\GlsXtrSetRecordCountAttribute{abbreviation}{1}

\renewcommand{\glslinkpresetkeys}{%
 \glsadd[format=glsignore,counter=chapter]{\gls-
label}}

\renewcommand*{\glsxtrrecordtriggervalue}[1]{%
 \GlsXtrLocationRecordCount{#1}{chapter}{\the-
chapter}%
}
\begin{document}
\chapter{First}
\gls{html}. \gls{html}. \gls{html}. \gls{ssi}.

\chapter{Second}
\gls{html}. \gls{ssi}. \gls{ssi}.  \gls{xml}.

\renewcommand*{\printunsrtglossaryentryprocesshook}
[1]{%
 \glsxtrifhasfield*{location}{#1}
 {}{\printunsrtglossaryskipentry}%
}
\printunsrtglossaries
\end{document}
```

If the document is in a file called `myDoc.tex` then the document build is:

```
pdflatex myDoc
bib2gls --record-count-unit myDoc
pdflatex myDoc
```

Example 160: Using bib2gls: unit record counting



11.5.2. Mini-Glossaries

Record counting doesn't have to be used with the \rgls set of commands. When bib2gls writes the code to the glstex file to save the record counting information, it does it with helper commands that it provides in the glstex file:

```
\bibglssettotalrecordcount{⟨entry-label⟩}{⟨value⟩}
```

This sets the total record count and is defined in the glstex file as:

```
\providecommand*{\bibglssettotalrecordcount}[2]{%
 \GlsXtrSetField{#1}{recordcount}{#2}%
}
```

```
\bibglssetrecordcount{⟨entry-label⟩}{⟨counter⟩}{⟨value⟩}
```

This sets the total for the given counter and is defined as:

```
\providecommand*{\bibglssetrecordcount}[3]{%
 \GlsXtrSetField{#1}{recordcount.#2}{#3}%
}
```

The following is only available with --record-count-unit:

```
\bibglssetlocationrecordcount{⟨entry-label⟩}{⟨counter⟩}
{⟨location⟩}{⟨value⟩}
```

This sets the total for the given location and is defined as:

```
\providecommand*{\bibglssetlocationrecordcount}[4]
{%
 \GlsXtrSetField{#1}{recordcount.#2.\glsxtrdetok-
location{#3}}{#4}%
}
```

By defining one of more of these commands before the glstex file is input, it's possible to pick up the information, without the need to iterate over all entries later. For example, the following will create a mini-glossary for each particular location and populate it with entries that have a record for that location.

```
\newcommand*{\bibglssetlocationrecordcount}[4]{%
 \GlsXtrSetField{#1}{recordcount.#2.#3}{#4}%
 \provideignoredglossary{minigloss.#2.#3}%
 \glsxtrcopytoglossary{#1}minigloss.#2.#3%
}
```

I've omitted \glsxtrdetoklocation for clarity and because I'm confident the locations won't be problematic. The mini-glossary can then be displayed at the start of the chapter with:

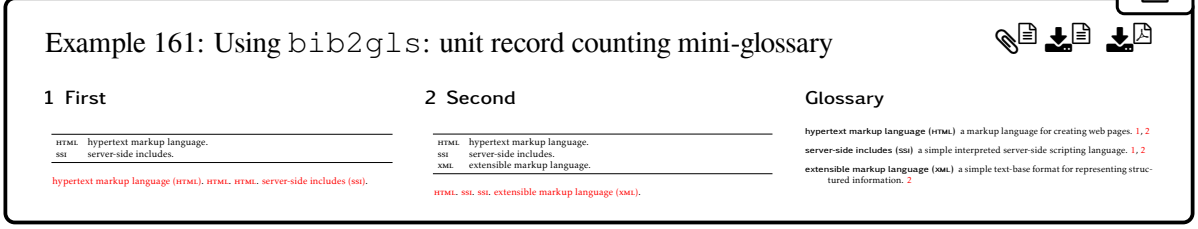
```
\printunsrtglossary[type=minigloss.chapter.\theH-
chapter]
```

The previous example can be altered to strip the \rgls commands and instead add a mini-glossary at the start of each chapter (the redefinition of \glslinkpresetkeys remains to ensure there are locations with the chapter counter). I've also provided a command to make it easier to display the mini-glossaries:

```
\newcommand{\minigloss}{%
 \printunsrtglossary*[style=abbr-short-long,type=
minigloss.chapter.\theHchapter,groups=false,target=
false]%
 {\renewcommand{\glossarysection}[2][]{}%
  \renewcommand{\glslongextraShortLongTabularHeader}
{\toprule}%
 }}
```

The document build is the same.

Example 161: Using bib2gls: unit record counting mini-glossary



11.6. The glossaries-extra-bib2gls package

\usepackage{**glossaries-extra-bib2gls**}
automatically loaded with \usepackage[record]{glossaries-extra}
or \usepackage[record=nameref]{glossaries-extra}

The package options record=only (or simply record) and record=nameref automatically loads the supplementary package glossaries-extra-bib2gls, which provides some commands that are specific to bib2gls, in particular to sorting and location lists which aren't relevant with record=hybrid.

If glossaries-extra-bib2gls is loaded via the record package option then the check for associated language files (see §15) will also search for the existence of glossariesxtr-⟨*script*⟩.ldf for each document dialect (where ⟨*script*⟩ is the four letter script identifier, such as Latn).

11.6.1. Displaying Glossaries

Glossaries are displayed with the "unsrt" family of commands (see §8.4). Some styles, such as bookindex, are customized for use with bib2gls.

The following commands are shortcuts that use \printunsrtglossary. However, they are only defined if a corresponding package option has been set *before* glossaries-extra-bib2gls is loaded. This means that the options must be passed as a package option, not using \glossariesextrasetup, if the shortcut commands are required.

\printunsrtabbreviations[⟨*options*⟩]

This shortcut is provided if the abbreviations package option has been used. This is a shortcut for:

\printunsrtglossary[type=abbreviations,⟨*options*⟩]

\printunsrtacronyms[⟨*options*⟩]

This shortcut is provided if the acronyms (or acronym) package option has been used. This is a shortcut for:

\printunsrtglossary[type=\acronymtype,⟨*options*⟩]

```
\printunsrtsymbols[⟨options⟩]
```

This shortcut is provided if the `symbols` package option has been used. This is a shortcut for:

```
\printunsrtglossary[type=symbols,⟨options⟩]
```

```
\printunsrtnumbers[⟨options⟩]
```

This shortcut is provided if the `numbers` package option has been used. This is a shortcut for:

```
\printunsrtglossary[type=numbers,⟨options⟩]
```

```
\printunsrtindex[⟨options⟩]
```

This shortcut is provided if the `index` package option has been used. This is a shortcut for:

```
\printunsrtglossary[type=index,⟨options⟩]
```

11.6.2. Helper Commands for Resource Options

```
\glshex⟨hex⟩
```

This simply expands to `\string\u`⟨*hex*⟩, which is used to identify the Unicode character ⟨*hex*⟩ in the value of some resource options.

```
\glshashchar
```

This expands to a literal # character (similar to `\glsbackslash`).

```
\glscapturedgroup⟨n⟩
```

This simply expands to `\string\$`⟨*n*⟩ which is used to indicate the ⟨*n*⟩th captured group in a regular expression replacement in the value of some resource options (requires `bib2gls` v1.5+), such as `sort-replace`. For example:

```
sort-replace={{([a-zA-Z])\string\.}{\glscaptured-
group1}}
```

This removes a full stop that follows any of the characters a,…,z or A,…,Z.

Note that \glscapturedgroup isn't the same as the match group identifier \MGP.

If you have complex regular expressions or use assign-fields, you may find it more convenient to redefine \glsxtrresourceinit to use the following command:

```
\GlsXtrResourceInitEscSequences
```

\GlsXtrResourceInitEscSequences should not be used outside of the definition of \glsxtrresourceinit as the definitions will likely cause interference and are only intended as resource option instructions for bib2gls.

This command locally redefines escape sequences used in regular expressions so that they detokenize when they expand. This means that you won't need to use \string or \protect in front of them. The following commands are defined:

- \u⟨*hex*⟩ (Unicode character);
- General use: \cs{⟨*csname*⟩} (expands to detokenized \csname when writing to the aux file);
- For literal characters in regular expressions: \. \\ \/ \| \& \+ \< \< * \$ \^ \~ \(\) \[\] \" \- \? \: \#
- Special markup for use as instructions or keywords in assign-fields: \MGP \LEN \CAT \TRIM \CS \INTERPRET \LABELIFY \LABELIFYLIST \NULL \IN \NIN \PREFIXOF \NOTPREFIXOF \SUFFIXOF \NOTSUFFIXOF \LC \UC \FIRSTLC \FIRSTUC \TITLE.

For example, with

```
\renewcommand{\glsxtrresourceinit}{%
  \GlsXtrResourceInitEscSequences
}
```

then the earlier example can be written more compactly as:

```
sort-replace={{([a-zA-Z])\.}{\$1}}
```

A convenient shortcut for use in the `entry-type-aliases` setting:

```
\GlsXtrBibTeXEntryAliases
```

This provides aliases for BibTeX's standard entry types (such as `@article` and `@book`) to `bib2gls`'s `@bibtexentry` entry type (requires `bib2gls` v1.4+).

You may also want to provide storage keys for BibTeX's standard fields rather than having to alias them all. This can be done with:

```
\GlsXtrProvideBibTeXFields
```

This defines each BibTeX field, such as `author`, as a glossary entry key:

```
\glsaddstoragekey{address}{}{\glsxtrbibaddress}%
 \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
 \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}
%
 \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
 \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
 \glsaddstoragekey{howpublished}{}{\glsxtrbibhow-
published}%
 \glsaddstoragekey{institution}{}{\glsxtrbib-
institution}%
 \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
 \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
 \glsaddstoragekey{note}{}{\glsxtrbibnote}%
 \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
 \glsaddstoragekey{organization}{}{\glsxtrbib-
organization}%
 \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
 \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}
%
 \glsaddstoragekey{school}{}{\glsxtrbibschool}%
 \glsaddstoragekey{series}{}{\glsxtrbibseries}%
 \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
 \glsaddstoragekey{bibtextype}{}{\glsxtrbibtype}%
 \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
```

This command should be placed before the first \GlsXtrLoadResources.

BibTeX's type field clashes with the glossaries package's type key, so this command provides the key bibtextype instead. You can alias it with field-aliases={type=bibtextype} in the resource options.

11.6.2.1. Custom Sort

There are many locale alphabetical rules provided with bib2gls, such as sort=de-1996 for German new orthography. However, it may be that your particular locale isn't supported, or you want a rule that covers multiple scripts or non-alphabetic symbols.

The sort=custom setting combined with sort-rule provides a way to define your own sort rule. For example, suppose I have a file called animals.bib that contains:

```
@index{bee}
@index{lion}
@index{ant}
@index{cow}
@index{goose}
@index{zebu}
@index{egret}
@index{elk}
@index{llama}
@index{lynx}
@index{bat}
```

Here's a very limited rule that only recognises five letters:

```
\usepackage[record,nostyles,stylemods=
bookindex,style=bookindex]
{glossaries-extra}

\newcommand{\bibglssetlastgrouptitle}[2]{%
 \glsxtrsetgrouptitle{#1#2}{Other}%
}
\GlsXtrLoadResources[src={animals},selection=all,
 sort=custom,sort-rule=
{ < a,A < b,B < e,E < l,L < ll,Ll,LL < z,Z}]
```

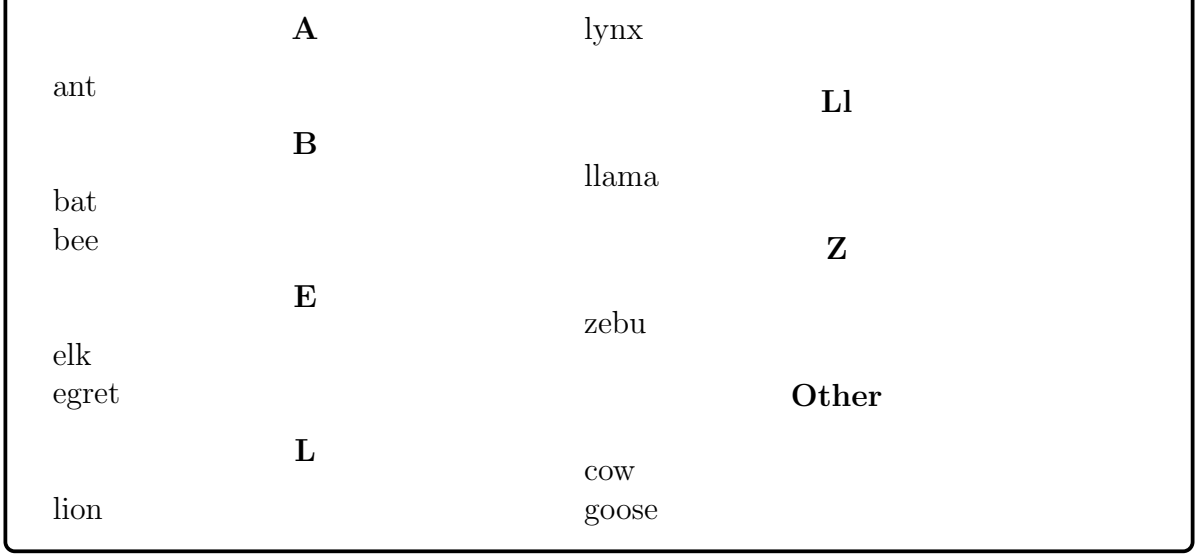
```
\begin{document}
\printunsrtglossaries
\end{document}
```

Any characters that aren't included in the rule (such as "c" and "g") are placed at the end. I've defined \bibglssetlastgrouptitle to label that final group of characters "Other". If the document is in a file called myDoc.tex, the build process is:

```
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
```

The result is:

Example 162: Using bib2gls: simple custom sort rule



Note that "egret" has been placed after "elk". This is because "l" is included in the rule but "g" isn't. Whereas "lynx" comes before "llama" because there's a separate "ll" group after the "l" group.

The commands listed below provide common rule blocks for use in the sort-rule resource option. If you want a rule for a specific locale, you can provide similar commands in a file called glossariesxtr-⟨*tag*⟩.ldf, where ⟨*tag*⟩ identifies the dialect, locale, region or root language. See the description of \IfTrackedLanguageFileExists in the tracklang documentation for further details. If this file is on TEX's path and the tracklang package (automatically loaded by glossaries) detects that the document has requested that language

or locale, then the file will automatically be loaded. For example, if you want to provide a rule block for Welsh, then create a file called `glossariesxtr-welsh.ldf` that contains:

```
\ProvidesGlossariesExtraLang{welsh}[2018/02/23 v1.0]

\@ifpackageloaded{glossaries-extra-bib2gls}
{
  \newcommand{\glsxtrWelshRules}{%
   \glsxtrLatinA
   \string<b,B
   \string<c,C
   \string<ch,CH
   \string<d,D
   \string<dd,DD
   \string<\glsxtrLatinE
   \string<f,F
   \string<ff,FF
   \string<g,G
   \string<ng,NG
   \string<\glsxtrLatinH
   \string<\glsxtrLatinI
   \string<j,J
   \string<\glsxtrLatinL
   \string<ll,Ll,LL
   \string<\glsxtrLatinM
   \string<\glsxtrLatinN
   \string<\glsxtrLatinO
   \string<\glsxtrLatinP
   \string<ph,PH
   \string<r,R
   \string<rh,RH
   \string<\glsxtrLatinS
   \string<\glsxtrLatinT
   \string<th,TH
   \string<u,U
   \string<w,W
   \string<y,Y
  }
}
{}% glossaries-extra-bib2gls.sty not loaded
```

(The use of `\string` is in case the < character has been made active.) You can provide more than one rule-block per local, to allow for loanwords or foreign words. For example, you could

provide \glsxtrWelshIRules, \glsxtrWelshIIRules etc.

If the rules are for a particular script (independent of language or region) then they can be provided in a file given by glossariesxtr-⟨*script*⟩.ldf instead. For example, the file glossariesxtr-Cyrl.ldf could contain:

```
\ProvidesGlossariesExtraLang{Cyrl}[2018/02/23 v1.0]
\newcommand*{\glsxtrGeneralCyrillicIRules}{%
  % Cyrillic rules
}
\newcommand*{\glsxtrGeneralCyrillicIIRules}{%
  % an alternative set of Cyrillic rules
}
```

Remember that the required document language scripts need to be tracked through the tracklang package, in order for these files to be automatically loaded. This essentially means ensuring you load the appropriate language package before tracklang is loaded by the base glossaries package or any other package that uses it. See the tracklang documentation for further details.

Alternatively, if the rules are specific to a subject rather than a region or language, then you can provide a supplementary package. For example, if you have a package called, say, mapsymbols that provides map symbols, then the file mapsymbols.sty might contain:

```
\ProvidesPackage{mapsymbols}
% some package or font loading stuff here to provide
% the appropriate symbols
\newcommand{\Stadium}{...}
\newcommand{\Battlefield}{...}
\newcommand{\Harbour}{...}
% etc

% Provide a rule block:
\newcommand{\MapSymbolOrder}{%
 \glshex 2694 % crossed-swords 0x2694
 \string< \glshex 2693 % anchor 0x2693
 \string< \glshex 26BD % football 0x26BD
}
```

and the supplementary file mapsymbols.bib can provide the appropriate definitions for bib2gls:

```
@preamble{"\glsxtrprovidecommand{\Harbour}
{\char"2693}
\glsxtrprovidecommand{\Battlefield}{\char"2694}
\glsxtrprovidecommand{\Stadium}{\char"26BD}"}
```

Now both the preamble and rule block can be used in the resource set:

```
\usepackage{mapsymbols}% my custom package
\usepackage[record]{glossaries-extra}

\GlsXtrLoadResources[
 src={mapsymbols,% <--- my custom mapsymbols.bib
  entries% data in entries.bib
 },
 sort=custom,
 sort-rule={\glsxtrcontrolrules % control codes
 ;\glsxtrspacerules % space characters
 ;\glsxtrnonprintablerules % non-printable characters
 ;\glsxtrcombiningdiacriticrules % combining diacritics
 ,\glsxtrhyphenrules % hyphens
 <\glsxtrgeneralpuncrules % general punctuation
 <\glsxtrdigitrules % 0, ..., 9
 <\glsxtrfractionrules % fraction symbols
 <\MapSymbolOrder % <--- custom map symbols
 <\glsxtrMathItalicGreekIrules % math-greek symbols
 <\glsxtrGeneralLatinIrules % Latin letters
 }
]
```

(As before, you may need to use `\string` in front of characters like < if they have been made active.)

The following commands are provided by glossaries-extra-bib2gls. They should be separated by the rule separator characters `;` (semi-colon) or `,` (comma) or `&` (ampersand) or < (less than). See Java's RuleBasedCollator documentation for details of the rule syntax.

For example, the following will place the mathematical Greek symbols (`\alpha`, `\Alpha`, `\beta`, `\Beta` etc) in a block before Latin characters:

```
sort-rule={\glsxtrcontrolrules
 ;\glsxtrspacerules
 ;\glsxtrnonprintablerules
 ;\glsxtrcombiningdiacriticrules
```

```
 ,\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
 <\glsxtrMathItalicGreekIrules
 <\glsxtrGeneralLatinIVrules
 <\glsxtrLatinAA
 <\glsxtrLatinOslash
}
```

11.6.2.1.1. Non-Letters

Remember that the default break-point setting is `break-at=word`. This means that non-alphabetic characters will typically be stripped and you could end up with empty sort values. If you want punctuation characters sorted you will need to change this setting to `break-at=none`. If you still want to retain letter sorting for any words or phrases included in the list you can use `sort-replace` (see the `bib2gls` manual for further details of these options).

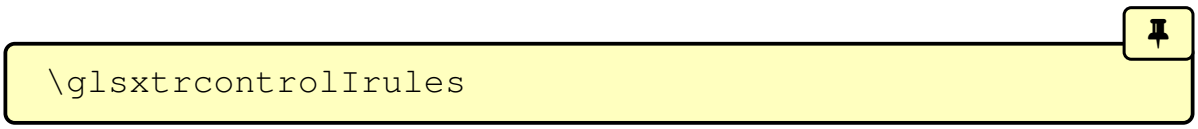
For example:

```
\GlsXtrLoadResources[
  sort=custom,
  sort-rule={
   \glsxtrIgnorableRules
   ;\glsxtrcombiningdiacriticrules
   <\glsxtrGeneralPuncRules
   <\glsxtrGeneralLatinIrules
  },
 break-at=none,
 sort-replace={{,? +}{\glshex001F}}
]
```

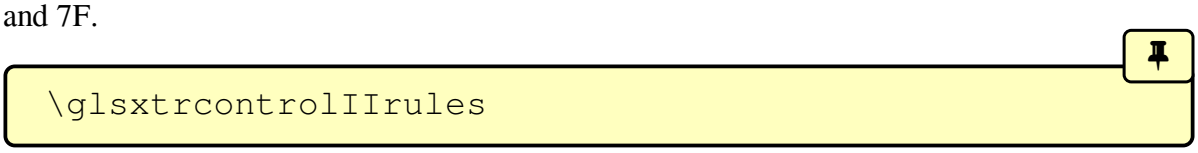
```
\glsxtrcontrolrules
```

These are control characters that are usually placed at the start of a rule in the ignored section. They typically won't occur in any sort values, but if they do they should normally be ignored.

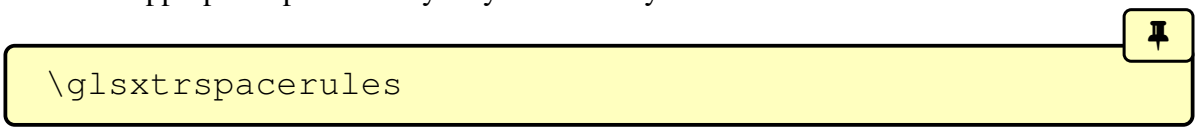
However, with `bib2gls` version 4.0+, the `--datatool-sort-markers` switch may be used to define the datatool-base marker commands according to the way they work within `\DTLsortwordlist`. This means that they can expand to control codes within the interpreter. Since they are normally ignored by language-sensitive sort rules, a custom rule will be needed.



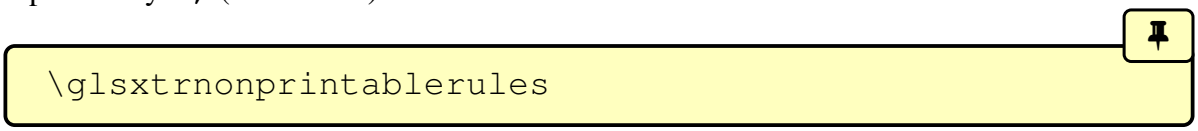
Expands as per \glsxtrcontrolrules but omits the control codes 0, 1C, 1D, 1E, 1F and 7F.



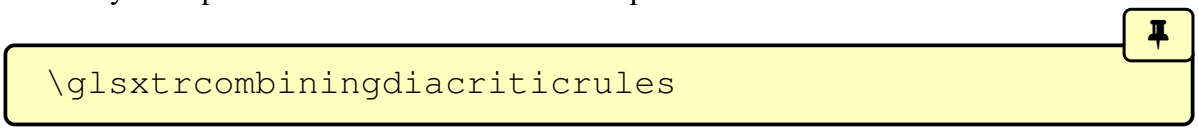
Expands to the ordered set of "information separator" control codes 1C, 1D, 1E, and 1F. Note that if you use \glsxtrcontrolIrules and \glsxtrcontrolIIrules instead of \glsxtrcontrolrules, you will also need to insert \glshex 0 and \glshex 7F in the appropriate place if they may occur in any of the sort values.



These are space characters. They typically come after the control characters with the two blocks separated by a ; (semi-colon).



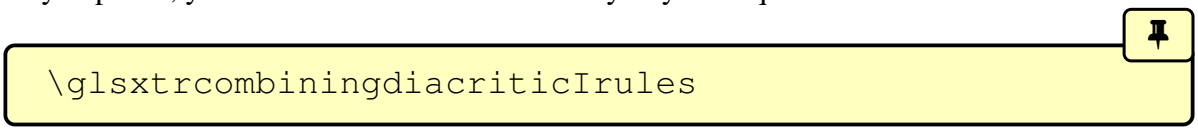
These are non-printable characters (BOM, tabs, line feed and carriage return). They typically come after the spaces separated by a ; (semi-colon). These characters aren't checked for by bib2gls when it determines whether or not to use the interpreter, so a TAB or newline character may end up in the sort value if it wasn't interpreted.



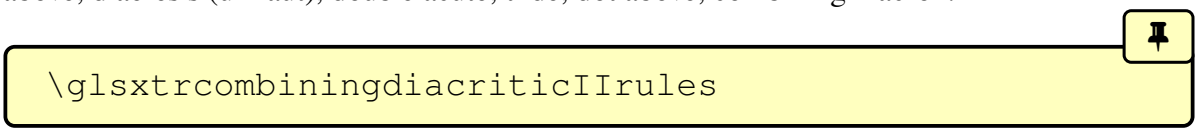
These are combining diacritic marks which typically follow the space and non-printable blocks (separated by a semi-colon). This command is defined in terms of sub-block commands:

```
\newcommand*{\glsxtrcombiningdiacriticrules}{%
 \glsxtrcombiningdiacriticIrules\string;
 \glsxtrcombiningdiacriticIIrules\string;
 \glsxtrcombiningdiacriticIIIrules\string;
 \glsxtrcombiningdiacriticIVrules
}
```

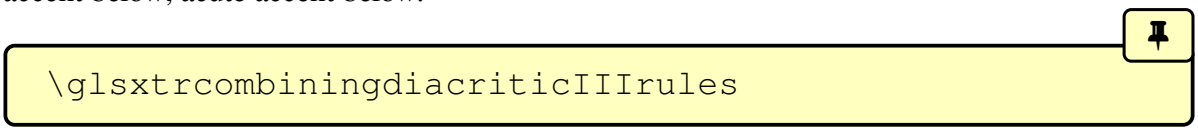
If you prefer, you can use the sub-blocks directly in your required ordered.



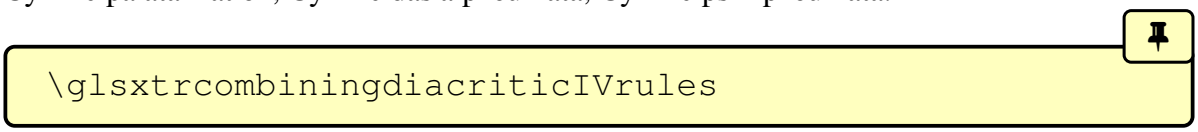
This contains the combining diacritics: acute, grave, breve, circumflex, caron, ring, vertical line above, diaeresis (umlaut), double acute, tilde, dot above, combining macron.



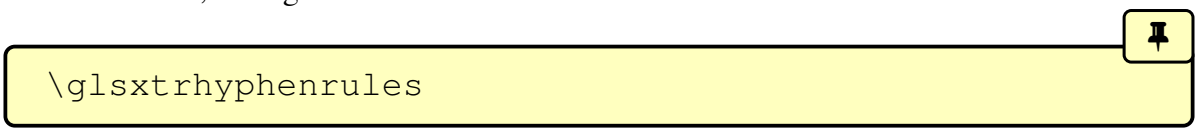
This contains the combining diacritics: short solidus overlay, cedilla, ogonek, dot below, low line, overline, hook above, double vertical line above, double grave accent, candrabindu, inverted breve, turned comma above, comma above, reversed comma above, comma above right, grave accent below, acute accent below.



This contains the combining diacritics: left tack below, right tack below, left angle above, horn, left half ring below, up tack below, down tack below, plus sign below, minus sign below, palatalized hook below, retroflex hook below, diaresis below, ring below, comma below, vertical line below, bridge below, inverted double arch below, caron below, circumflex accent below, breve below, inverted breve below, tilde below, macron below, double low line, tilde overlay, short stroke overlay, long stroke overlay, long solidus overlay, right half ring below, inverted bridge below, square below, seagull below, x above, vertical tilde, double overline, Greek perispomeni, Greek dialytika tonos, Greek ypogegrammeni, double tilde, double inverted breve, Cyrillic titlo, Cyrillic palatalization, Cyrillic dasia pneumata, Cyrillic psili pneumata.



This contains the combining diacritics: left harpoon above, right harpoon above, long vertical line overlay, short vertical line overlay, anticlockwise arrow above, clockwise arrow above, left arrow above, right arrow above, ring overlay, clockwise ring overlay, anticlockwise ring overlay, three dots above, four dots above, enclosing circle, enclosing square, enclosing diamond, enclosing circle backslash, left right arrow above.



This contains hyphens (including the minus sign 0x2212). This rule block typically comes after the diacritic rules separated by a comma. As from version v1.56, the hyphen rules are now divided into two sub-sets:

```
\newcommand*{\glsxtrhyphenrules}{%
 \glsxtrhyphenIrules;
 \glsxtrminusrules
}
```

```
\glsxtrhyphenIrules
```

This first set consists of: ASCII hyphen/minus, soft hyphen, non-breaking hyphen, figure dash, en dash, em dash and horizontal bar, which are all considered equivalent. The minus sign (0x2212) is not included.

```
\glsxtrhyphenIIrules
```

An alternative to the first set of hyphen rules, this set has the same characters but lists them in ascending order. That is, they are not considered equivalent.

The minus rules are in the sub-set:

```
\glsxtrminusrules
```

This set consists of: minus sign (0x2212), superscript minus and subscript minus. (Not the ASCII hyphen/minus.) If you don't want the minus signs included in the hyphen rules, use `\glsxtrhyphenIrules` (or `\glsxtrhyphenIIrules`) instead of `\glsxtrhyphenrules`. If you want the hyphens and minus signs between the percent and plus sign, use `\glsxtrgeneralpuncIIIrules` instead of `\glsxtrgeneralpuncIIrules`.

```
\glsxtrgeneralpuncrules
```

This contains punctuation characters. This rule block typically comes after the hyphen rules separated by a less than (<). As with the combining diacritics, this command is defined in terms of sub-blocks which may be used directly instead if a different order is required:

```
\newcommand*{\glsxtrgeneralpuncrules}{%
  \glsxtrgeneralpuncIrules
  \string<\glsxtrcurrencyrules
  \string<\glsxtrgeneralpuncIIrules
}
```

```
\glsxtrgeneralpuncIrules
```

This is defined as:

```
\newcommand*{\glsxtrgeneralpuncIrules}{%
 \glsxtrgeneralpuncmarksrules
 \string<\glsxtrgeneralpuncaccentsrules
 \string<\glsxtrgeneralpuncquoterules
 \string<\glsxtrgeneralpuncbracketrules
 \string<\glsxtrgeneralpuncsignrules
}
```

```
\glsxtrgeneralpuncmarksrules
```

This contains: underscore, macron, comma, semi-colon, colon, exclamation mark, inverted exclamation mark, question mark, inverted question mark, solidus, full stop.

`\glsxtrgeneralpuncdotrules` This contains dotted punctuation marks in the General Punctuation block.

```
\glsxtrgeneralpuncaccentsrules
```

This contains: acute accent, grave accent, circumflex accent, diaeresis, tilde, middle dot, cedilla.

```
\glsxtrgeneralpuncquoterules
```

This contains: straight apostrophe, straight double quote, left guillemet, right guillemet.

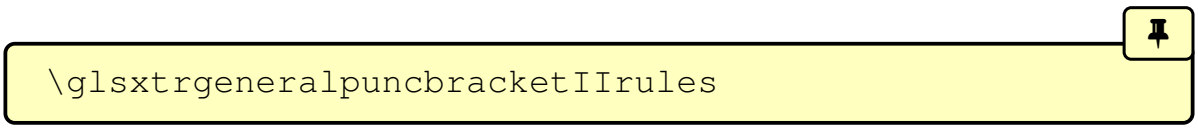
```
\glsxtrgeneralpuncbracketrules
```

By default this just expands to the first subset of bracket rules `\glsxtrgeneralpuncbracketIrules` but may be redefined to include extra sets. For example:

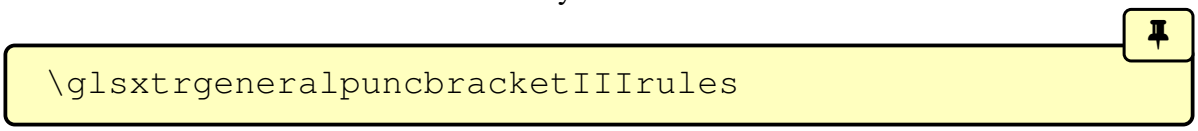
```
\renewcommand{\glsxtrgeneralpuncbracketrules}%
 \glsxtrgeneralpuncbracketIrules
 <\glsxtrgeneralpuncbracketIIrules
```

```
\glsxtrgeneralpuncbracketIrules
```

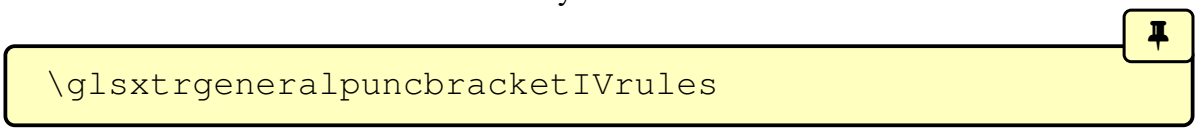
This contains: left parenthesis, right parenthesis, left square bracket, right square bracket, left curly bracket, right curly bracket, left angle bracket, right angle bracket.



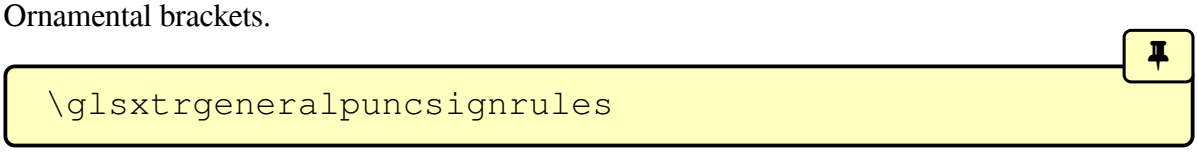
Brackets from miscellaneous mathematical symbols A.



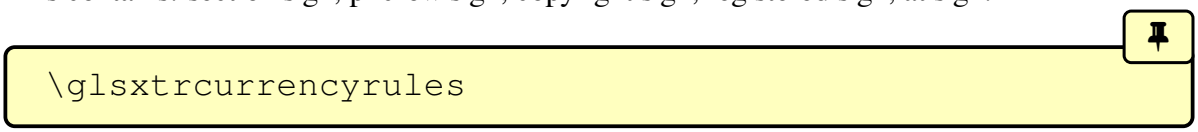
Brackets from miscellaneous mathematical symbols B.



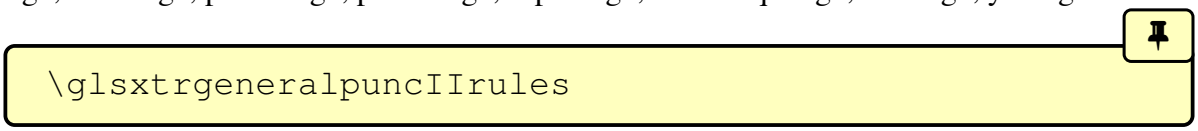
Ornamental brackets.



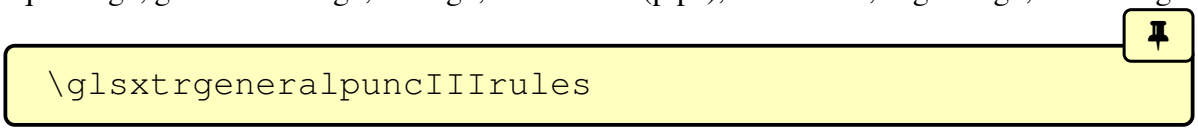
This contains: section sign, pilcrow sign, copyright sign, registered sign, at sign.



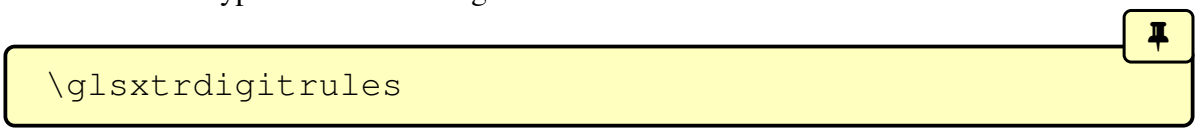
This sub-block contains some currency symbols: currency sign, Thai currency symbol baht, cent sign, colon sign, cruzeiro sign, dollar sign, dong sign, euro sign, French franc sign, lira sign, mill sign, naira sign, peseta sign, pound sign, rupee sign, new sheqel sign, won sign, yen sign.



This sub-block contains some other punctuation symbols: asterisk, backslash, ampersand, hash sign, percent sign, plus sign, plus-minus sign, division sign, multiplication sign, less-than sign, equals sign, greater-than sign, not sign, vertical bar (pipe), broken bar, degree sign, micron sign.



An alternative to `\glsxtrgeneralpuncIIrules`, this sub-block includes `\glsxtrhyphenIIrules` and `\glsxtrminusrules` between the percent and plus signs. This ensures that the hyphens and minus signs are distinct.



This rule block contains the Basic Latin digits (0, …, 9) and the subscript and superscript digits ($_0$ ^{0} etc) made equivalent to the corresponding Basic Latin digit. The digit block typically comes after the punctuation rules separated by a less than (<).

```
\glsxtrBasicDigitrules
```

This rule block contains just the Basic Latin digits (0, …, 9).

```
\glsxtrSubScriptDigitrules
```

This rule block contains just the subscript digits (₀ … ₉).

```
\glsxtrSuperScriptDigitrules
```

This rule block contains just the superscript digits (⁰ … ⁹).

```
\glsxtrfractionrules
```

This rule block contains vulgar fraction characters from the Unicode Number Forms block. The digit block typically comes after the digit rules separated by a less than (<).

```
\glsxtrIgnorableRules
```

A shortcut that expands to the ignorable rules:

```
\glsxtrcontrolrules
 ;\glsxtrspacerules
 ;\glsxtrnonprintablerules
```

```
\glsxtrGeneralInitRules
```

A shortcut that expands to common initial rules:

```
\glsxtrIgnorableRules
 ;\glsxtrcombiningdiacriticrules
 ;\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
```

Note that this includes the combining diacritic rules, which won't be appropriate for languages with accented characters.

```
\glsxtrGeneralPuncRules
```

A shortcut that expands to common punctuation rules:

```
\glsxtrgeneralpuncmarksrules
 <\glsxtrgeneralpuncdotrules
 <\glsxtrgeneralpuncaccentsrules
 <\glsxtrgeneralpuncquoterules
 <\glsxtrgeneralpuncbracketrules
 <\glsxtrgeneralpuncsignrules
 <\glsxtrcurrencyrules
 <\glsxtrgeneralpuncIIIrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
```

There are a number of Latin rule blocks. Some of these included extended characters or ligatures (such as ß or œ) but they don't include accented characters. If you require a Latin rule block that includes accented characters, digraphs, trigraphs or other extended characters, then it's best to provide similar commands in a `glossariesxtr-`⟨*tag*⟩`.ldf` file for the particular language or region.

11.6.2.1.2. Latin Letters

```
\glsxtrGeneralLatinIrules
```

This is just the basic (non-extended) Latin alphabet with the superscript and subscript Latin letters (^{a} $_{a}$ etc) treated as the equivalent basic Latin letter. (If you don't want the subscripts and superscripts included you can redefine `\glsxtrLatinA` etc to omit them.)

```
\glsxtrGeneralLatinIIrules
```

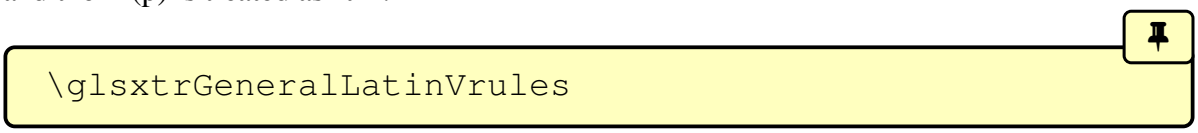
This is like `\glsxtrGeneralLatinIrules` but it includes eth (Ð) between "D" and "E" and eszett (ß) treated as "ss".

```
\glsxtrGeneralLatinIIIrules
```

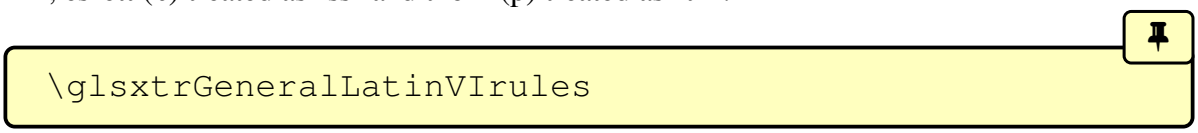
This is like `\glsxtrGeneralLatinIrules` but it includes eth (Ð) between "D" and "E" and eszett (ß) treated as "sz".

```
\glsxtrGeneralLatinIVrules
```

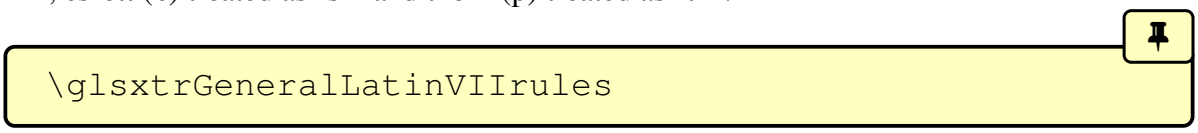
This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, ae-ligature (æ) is treated as “ae”, oe-ligature (œ) is treated as “oe”, eszett (ß) treated as “ss” and thorn (þ) is treated as “th”.



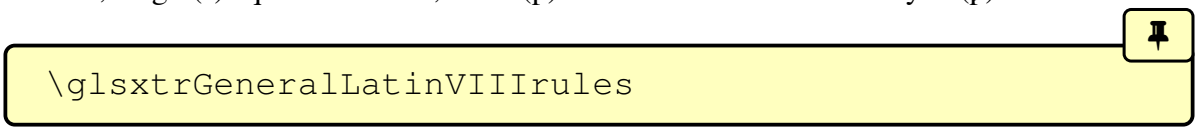
This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, eszett (ß) treated as “ss” and thorn (þ) treated as “th”.



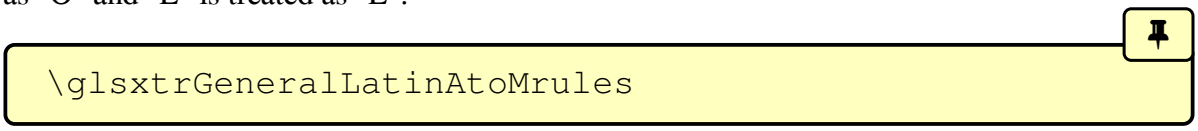
This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, eszett (ß) treated as “sz” and thorn (þ) treated as “th”.



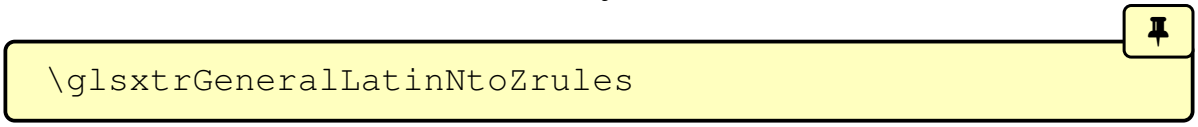
This is like \glsxtrGeneralLatinIrules but it includes ae-ligature (æ) between “A” and “B”, eth (Ð) between “D” and “E”, insular G (ᵹ) instead of “G”, oe-ligature (œ) between “O” and “P”, long s (ſ) equivalent to “s”, thorn (þ) between “T” and “U” and wynn (ƿ) instead of “W”.



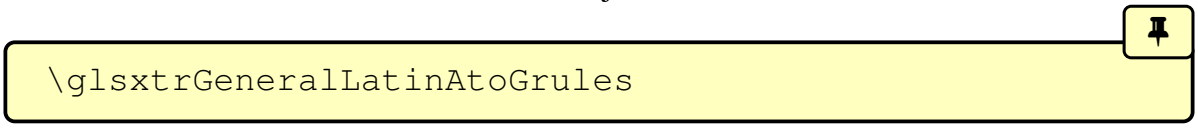
This is like \glsxtrGeneralLatinIrules but ae-ligature (æ) is treated as “ae”, oe-ligature (œ) is treated as “oe”, eszett (ß) treated as “ss”, thorn (þ) is treated as “th”, Ø is treated as “O” and “Ł” is treated as “L”.



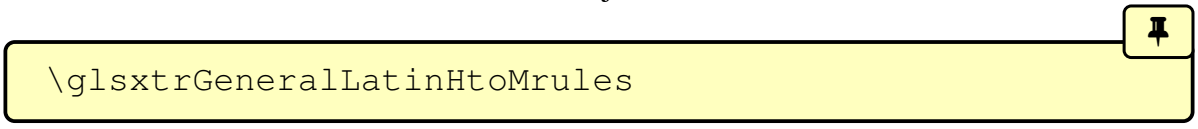
A mini-rule subset of General Latin I rules that just covers A–M.



A mini-rule subset of General Latin I rules that just covers N–Z.



A mini-rule subset of General Latin I rules that just covers A–G.



A mini-rule subset of General Latin I rules that just covers H–M.

```
\glsxtrGeneralLatinNtoSrules
```

A mini-rule subset of General Latin I rules that just covers N–S.

```
\glsxtrGeneralLatinTtoZrules
```

A mini-rule subset of General Latin I rules that just covers T–Z.

```
\glsxtrLatinA
```

A mini-rule that just covers “A” but includes the sub- and superscript A.

```
\glsxtrLatinE
```

A mini-rule that just covers “E” but includes the subscript E.

```
\glsxtrLatinH
```

A mini-rule that just covers “H” but includes the subscript H.

```
\glsxtrLatinK
```

A mini-rule that just covers “K” but includes the subscript K.

```
\glsxtrLatinI
```

A mini-rule that just covers “I” but includes the subscript I.

```
\glsxtrLatinM
```

A mini-rule that just covers “M” but includes the subscript M.

```
\glsxtrLatinN
```

A mini-rule that just covers “N” but includes the sub- and superscript N.

```
\glsxtrLatinO
```

A mini-rule that just covers “O” but includes the sub- and superscript O.

```
\glsxtrLatinP
```

A mini-rule that just covers “P” but includes the subscript P.

```
\glsxtrLatinS
```

A mini-rule that just covers “S” but includes the subscript S.

```
\glsxtrLatinT
```

A mini-rule that just covers “T” but includes the subscript T.

```
\glsxtrLatinX
```

A mini-rule that just covers “X” but includes the subscript X.

```
\glsxtrLatinEszettSs
```

A mini-rule that just covers eszett (ß) and makes “fs” (long s followed by short s) equivalent to “ß”. (This is used in the above blocks that treat “ß” as “ss”.)

```
\glsxtrLatinEszettSz
```

A mini-rule that just covers eszett (ß) and makes “fz” (long s followed by z) equivalent to “ß”. (This is used in the above blocks that treat “ß” as “sz”.)

```
\glsxtrLatinEth
```

A mini-rule for eth (Ð, ð) so you don’t need to remember the Unicode values.

```
\glsxtrLatinThorn
```

A mini-rule for thorn (Þ, þ) so you don’t need to remember the Unicode values.

```
\glsxtrLatinAELigature
```

A mini-rule for ae-ligature (Æ, æ) so you don’t need to remember the Unicode values.

```
\glsxtrLatinOELigature
```

A mini-rule for oe-ligature (Œ, œ) so you don't need to remember the Unicode values.

```
\glsxtrLatinOslash
```

A mini-rule for o-slash (Ø, ø) so you don't need to remember the Unicode values.

```
\glsxtrLatinLslash
```

A mini-rule for l-slash (Ł, ł) so you don't need to remember the Unicode values.

```
\glsxtrLatinWynn
```

A mini-rule for wynn (Ƿ, ƿ) so you don't need to remember the Unicode values.

```
\glsxtrLatinInsularG
```

A mini-rule for insular-G (Ȝ, ȝ) so you don't need to remember the Unicode values.

```
\glsxtrLatinSchwa
```

A mini-rule for schwa (Ə, ə, ᵊ) so you don't need to remember the Unicode values. (Not used in any of the provided Latin rule blocks described above.)

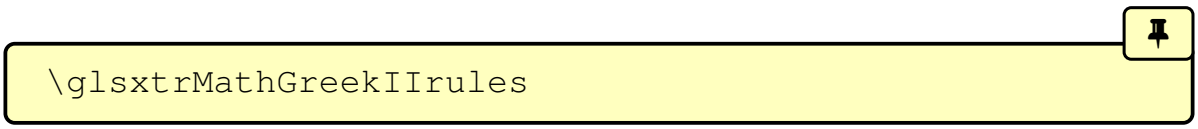
```
\glsxtrLatinAA
```

A mini-rule for "a with ring above" (Å, å) so you don't need to remember the Unicode values. (Not used in any of the provided Latin rule blocks described above.)

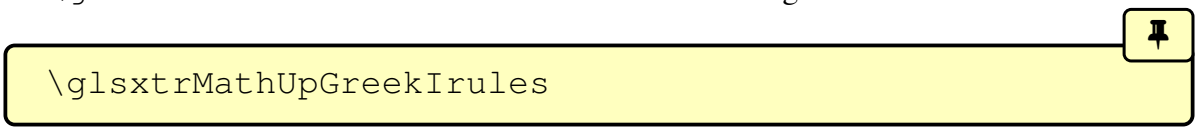
11.6.2.1.3. Math Greek

```
\glsxtrMathGreekIrules
```

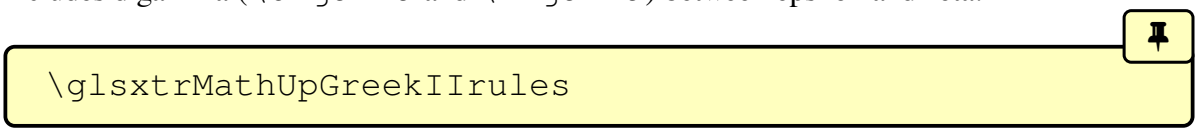
A rule block for mathematical Greek (`\alpha`, `\beta` etc) and upright Greek (`\upalpha`, etc, from the upgreek package) characters that includes digamma (`\digamma` and `\Digamma`) between epsilon and zeta. The upright and italic versions are gathered together into the same letter group.



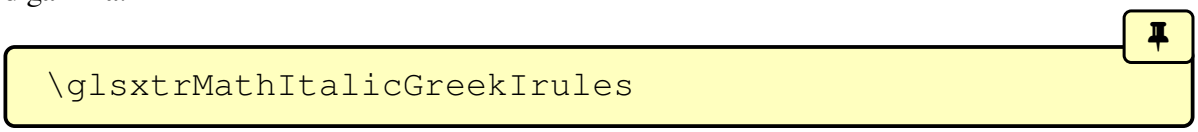
As \glsxtrMathGreekIrules but doesn't include digamma.



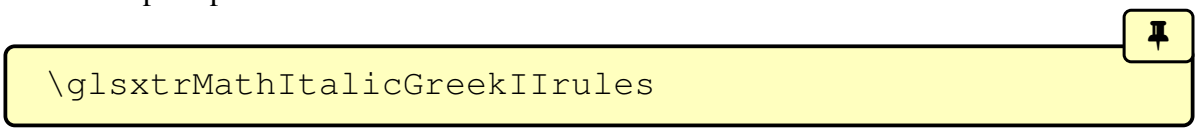
A rule block for upright Greek (\upalpha, etc, from the upgreek package) characters that includes digamma (\digamma and \Digamma) between epsilon and zeta.



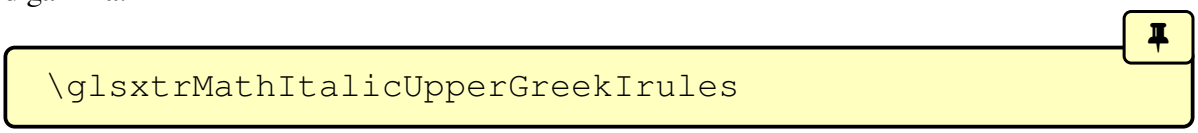
A rule block for upright Greek (\upalpha, etc, from the upgreek package) that doesn't include digamma.



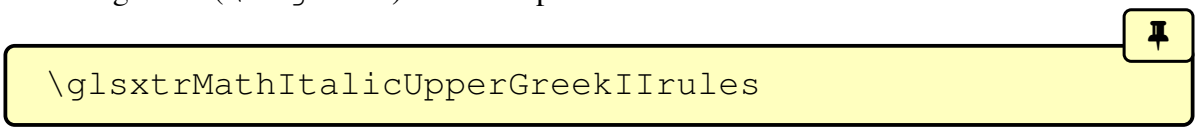
A rule block for mathematical Greek (\alpha, \Alpha, etc) characters that includes digamma (\digamma and \Digamma) between epsilon and zeta. Note that even though the uppercase \Delta etc are actually rendered upright by LaTeX, bib2gls's interpreter treats them as italic to help keep them close to the lowercase versions.



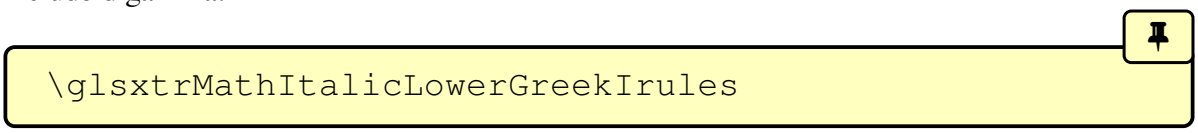
A rule block for mathematical Greek (\alpha, \Alpha, etc) characters that doesn't include digamma.



A rule block for uppercase mathematical Greek (\Alpha, \Beta, etc) characters that includes digamma (\Digamma) between epsilon and zeta.



A rule block for uppercase mathematical Greek (\Alpha, \Beta, etc) characters that doesn't include digamma.



A rule block for lowercase mathematical Greek (\alpha, \beta, etc) characters that includes digamma (\digamma) between epsilon and zeta.

```
\glsxtrMathItalicLowerGreekIIrules
```

A rule block for lower case mathematical Greek (\alpha, \beta, etc) characters that doesn't include digamma.

Additionally, there are commands that just cover the uppercase and lowercase forms of a special Greek character (\Upalpha, \upalpha etc and \Alpha, \alpha etc): \glsxtrUpAlpha, \glsxtrMathItalicAlpha, \glsxtrUpBeta, \glsxtrMathItalicBeta, \glsxtrUpGamma, \glsxtrMathItalicGamma, \glsxtrUpDelta, \glsxtrMathItalicDelta, \glsxtrUpEpsilon, \glsxtrMathItalicEpsilon, \glsxtrUpDigamma, \glsxtrMathItalicDigamma, \glsxtrUpZeta, \glsxtrMathItalicZeta, \glsxtrUpEta, \glsxtrMathItalicEta, \glsxtrUpTheta, \glsxtrMathItalicTheta, \glsxtrUpIota, \glsxtrMathItalicIota, \glsxtrUpKappa, \glsxtrMathItalicKappa, \glsxtrUpLambda, \glsxtrMathItalicLambda, \glsxtrUpMu, \glsxtrMathItalicMu, \glsxtrUpNu, \glsxtrMathItalicNu, \glsxtrUpXi, \glsxtrMathItalicXi, \glsxtrUpOmicron, \glsxtrMathItalicOmicron, \glsxtrUpPi, \glsxtrMathItalicPi, \glsxtrUpRho, \glsxtrMathItalicRho, \glsxtrUpSigma, \glsxtrMathItalicSigma, \glsxtrUpTau, \glsxtrMathItalicTau, \glsxtrUpUpsilon, \glsxtrMathItalicUpsilon, \glsxtrUpPhi, \glsxtrMathItalicPhi, \glsxtrUpChi, \glsxtrMathItalicChi, \glsxtrUpPsi, \glsxtrMathItalicPsi, \glsxtrUpOmega, and \glsxtrMathItalicOmega.

Additionally, there are commands for math italic partial differential ∂ (0x1D715) \glsxtrMathItalicPartial and nabla ∇ (0x1D6FB) \glsxtrMathItalicNabla.

11.6.3. Commands Used Within Resource Files

A glstex file will typically start with \glsnoexpandfields, since supporting field expansion is only necessary where entry definitions may be programmatically performed within a loop or some other command. (This may be suppressed.)

If you use the set-widest resource option, bib2gls v1.8+ will write the following command in the glstex file:

```
\glsxtrSetWidest{⟨type⟩}{⟨level⟩}
```

This sets the widest name for the given glossary type and hierarchical level. This supports both the alttree style and the styles provided by glossary-longextra, which need to know the widest name.

If bib2gls can't determine the widest name (typically because the name field consists of commands that aren't recognised by the interpreter) then bib2gls v1.8+ will write the following to the glstex file:

\glsxtrSetWidestFallback{⟨type⟩}{⟨level⟩}

{max depth}{list} Currently the maximum hierarchical depth ⟨*max depth*⟩ may only be 0 or 2. This command requires commands provided by the glossaries-extra-stylemods package with the alttree style enabled. In this case, it may be simpler to just use \glssetwidest.

The glstex files may also contain instances of \glsxtrprovidestoragekey to provide new glossary entry keys or \provideignoredglossary to provide glossaries to ensure they are defined.

The \bibgls... commands provided by bib2gls are defined in the glstex files with \providecommand. See the bib2gls manual for further details of those commands.

11.6.4. Hierarchy

The parent key identifies a parent entry (by its label), so determining if an entry has a parent can easily be achieved by testing if the parent field has been set (using \ifglshasparent or a command like \glsxtrifhasfield).

The other way around, testing if an entry has any children, is much harder. The base glossaries package provides \ifglshaschildren, which is very inefficient as it has to iterate over all entries in the entry's glossary to determine which ones have a matching parent field.

It's much easier with bib2gls, as there are resource options available to save this information. The save-child-count option saves the child count for each entry in the childcount internal field. This total only includes the child entries that have been selected by the resource set.

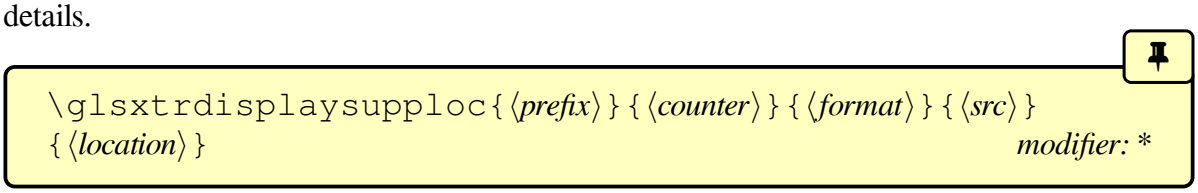
With glossaries v4.59+ and datatool v3.0+, the pre-sort function used by \printnoidxglossary may also locally set the childcount field for entries that have listed children. There is, however, a slight difference as, in this case, childcount isn't set for entries that don't have listed children, and the field is only locally set within the scope of the glossary.

\GlsXtrIfHasNonZeroChildCount{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
*modifier: **

A shortcut that uses \GlsXtrIfFieldNonZero to test if the value supplied in the childcount field is non-zero. The starred form uses the starred form of \GlsXtrIfFieldNonZero.

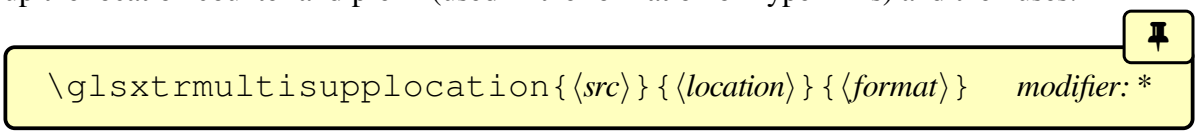
11.6.5. Supplemental Locations

Locations in external documents can be manually added by explicitly setting thevalue when an entry is indexed. However, this is a bit inconvenient so bib2gls provides a way to do this automatically. Both the main document and the supplemental document need to use the record option, and the entries provided via src in the main document must have the same labels as those in the supplementary document. The supplemental document is identified by the supplemental-locations resource option. See the bib2gls manual for further details.



\glsxtrdisplaysupploc{⟨*prefix*⟩}{⟨*counter*⟩}{⟨*format*⟩}{⟨*src*⟩}
{⟨*location*⟩} *modifier: **

This is used by bib2gls version v1.7+ for supplemental locations, instead of using \glsxtrsupphypernumber with the externallocation attribute. This command sets up the location counter and prefix (used in the formation of hyperlinks) and then uses:



\glsxtrmultisupplocation{⟨*src*⟩}{⟨*location*⟩}{⟨*format*⟩} *modifier: **

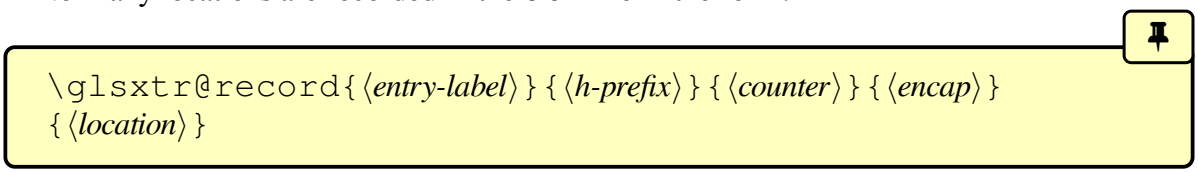
to format the actual location (with an external hyperlink, if supported). The ⟨*format*⟩ (the original location encap) is ignored by default.

11.6.6. Nameref Records

Nameref records include the current title and hypertarget. These records are enabled with record=nameref and provide a more reliable way of saving the location hypertarget, which can't be obtained with makeindex or xindy.

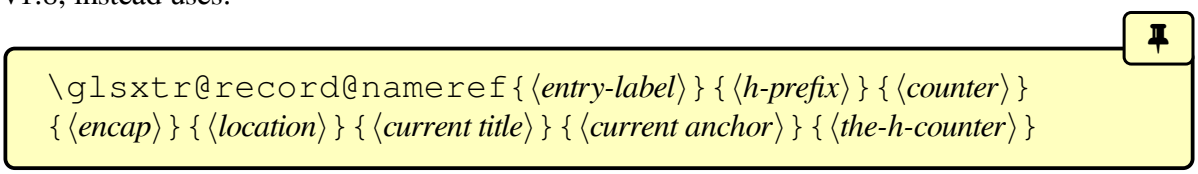
This option is best used with counter=chapter or counter=section if you want the title included in the location list. If the indexing counter is the default page, only the location number is shown. Similarly for counter=equation (or equations=true).

Normally locations are recorded in the aux file in the form:



\glsxtr@record{⟨*entry-label*⟩}{⟨*h-prefix*⟩}{⟨*counter*⟩}{⟨*encap*⟩}
{⟨*location*⟩}

This is similar to the command used with \makenoidxglossaries and provides the same information about the location and how to form the hypertarget as is passed to makeindex and xindy. The record=nameref option, which requires at least bib2gls v1.8, instead uses:



\glsxtr@record@nameref{⟨*entry-label*⟩}{⟨*h-prefix*⟩}{⟨*counter*⟩}
{⟨*encap*⟩}{⟨*location*⟩}{⟨*current title*⟩}{⟨*current anchor*⟩}{⟨*the-h-counter*⟩}

where ⟨*title*⟩ is obtained from \@currentlabelname and ⟨*href*⟩ is obtained from \@currentHref. These commands require hyperref. If they are undefined, ⟨*title*⟩ and ⟨*href*⟩ will be left empty and bib2gls will treat it as a regular record.

Be careful with this option as ⟨*href*⟩ will globally change on every instance of \refstepcounter but ⟨*title*⟩ won't necessarily change. It can therefore cause unexpected behaviour.

The final argument ⟨*hcounter*⟩ is obtained from \theH⟨*counter*⟩ which provides the partial target name associated with the location counter. With the original makeindex/xindy approach, it's not possible to include this information in the location, so the base glossaries package attempts to derive a prefix from which the ⟨*hcounter*⟩ value can be reconstituted by appending the prefix. Unfortunately, not all definitions of \theH⟨*counter*⟩ are in the form ⟨*prefix*⟩\the⟨*counter*⟩ (most notably the equation counter with chapters) so this can fail.

Since bib2gls is customized specifically for use with glossaries-extra, it's now possible to save ⟨*hcounter*⟩, so the recordnameref option does this. By providing both ⟨*href*⟩ and ⟨*hcounter*⟩, you can determine which target you would rather use. The default is to use ⟨*hcounter*⟩, which will take you to the place where the corresponding counter was incremented with \refstepcounter. However, you may choose to switch to using the ⟨*href*⟩ target, which will take you to the nearest target before the indexing took place.

With bib2gls v1.8+, normal locations are displayed using:

```
\glsnoidxdisplayloc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}
```

This is provided by the base glossaries package and is simply defined to do:

```
\setentrycounter[⟨prefix⟩]{⟨counter⟩}\csuse{⟨format⟩}{⟨location⟩}
```

Earlier versions of bib2gls only used this in the loclist field and explicitly used \setentrycounter in the location field followed by \⟨*format*⟩{location}, which follows the code that's created with the default makeindex setting. The \setentrycounter command sets up the prefix needed for \glshypernumber to reform the target name from the given location.

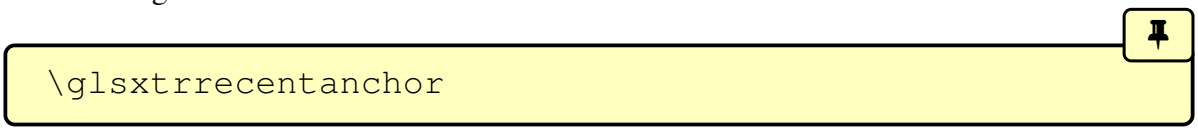
The locations identified by \glsxtr@record@nameref are written by bib2gls to the location list using:

```
\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}
{⟨location⟩}{⟨title⟩}{⟨href⟩}{⟨hcounter⟩}{⟨file⟩}
```

With normal internal locations, ⟨*file*⟩ will always be empty. With supplemental locations, ⟨*file*⟩ will be the external file reference. If hyperref hasn't been loaded, this command behaves like

\glsnoidxdisplayloc, so it will simply encapsulate the location with the given formatting command.

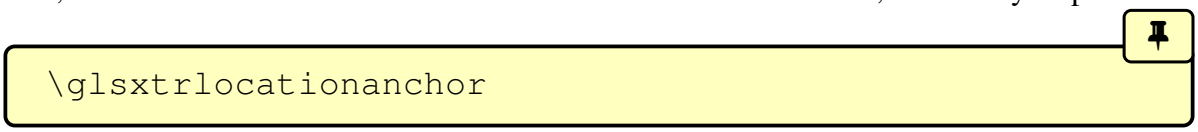
If hyperref has been loaded, then \glsxtrdisplaylocnameref will try to work out the appropriate hyperlink anchor and text. The ⟨*prefix*⟩ argument is redundant. It first defines the following commands:



\glsxtrrecentanchor

This expands to the ⟨*href*⟩ argument, which corresponds to the \@currentHref value at the time the location was recorded. If this is used as the anchor, the link will go to the most recent anchor before the record was created. This is more likely to match the given title, but won't necessarily match the corresponding counter.

For example, if the record was created with counter=section but occurred in a table caption, then \glsxtrrecentanchor and the title will correspond to the table caption. If you have defined \glsxtrsectionlocfmt to show the section number (see below), then this may cause some confusion if clicking on the section number leads to a table caption. However, it will lead to the closest location to where the record was created, which may be preferred.

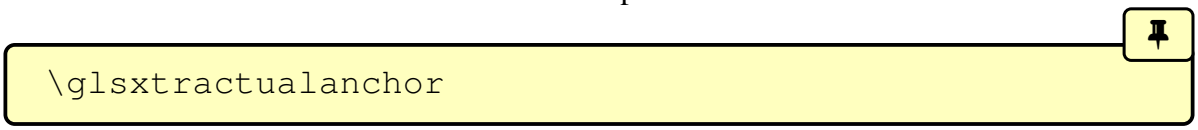


\glsxtrlocationanchor

This expands to the anchor that corresponds to the record's location counter, which is constructed from the ⟨*counter*⟩ and ⟨*hcounter*⟩ arguments.

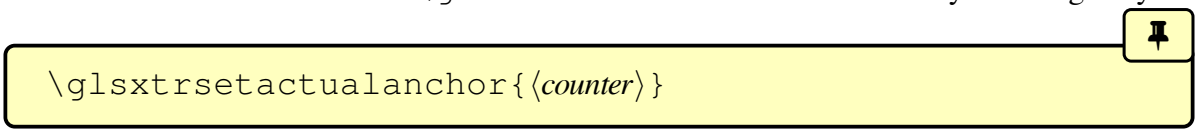
In the above example with the section counter, ⟨*hcounter*⟩ will be the value of \theHsection when the record was created, so the constructed anchor will be section.⟨*hcounter*⟩ which corresponds to the anchor at the start of the section. However, if you haven't defined \glsxtrsectionlocfmt, then the title will correspond to the table caption, which may be a little confusing.

The actual anchor used is obtained from the expansion of:



\glsxtractualanchor

This is initialised to the value of \glsxtrlocationanchor but may be changed by:



\glsxtrsetactualanchor{⟨*counter*⟩}

The argument is the counter name, which may be used to set choose an alternative anchor depending on the counter. This command does nothing by default, which means that \glsxtrlocationanchor will be used.

For example, to switch to \glsxtrrecentanchor if the counter is page:

```
\renewcommand{\glsxtrsetactualanchor}[1]{%
  \ifstrequal{#1}{page}
  \let\glsxtractualanchor\glsxtrrecentanchor  }
```

If you are using `indexcounter` then the recent anchor will be the one created by the wrglossary increment just before the indexing occurs. This means that with `counter=wrglossary`, the location anchor and recent anchor will be the same. With other counters, the recent anchor will be the closest anchor to the place where indexing occurred.

To allow for different formatting according to the counter name, `\glsxtrdisplaylocnameref` first checks for the existence of:

```
\glsxtr⟨counter⟩locfmt{⟨location⟩}{⟨title⟩}
```

If this command is defined, the location will be displayed using:

```
\glsxtrnamereflink{⟨format⟩}\glsxtr⟨counter⟩locfmt{⟨location⟩}
{⟨title⟩}{⟨href⟩}{⟨file⟩}
```

Note the above warning about the possible mismatch of the title with the anchor. For example, if the following is defined for the section counter:

```
\newcommand{\glsxtrsectionlocfmt}[2]{\S#1 (#2)}
```

then this could lead to the section number followed by the table caption. A more compact form that omits the title is better:

```
\newcommand{\glsxtrsectionlocfmt}[2]{\S#1}
```

The following location formats are predefined. The equation counter:

```
\glsxtrequationlocfmt{⟨location⟩}{⟨title⟩}
```

This simply expands to (⟨*location*⟩) since equations typically don't have a title, but are usually enclosed in parentheses.

```
\glsxtrwrglossarylocfmt{⟨location⟩}{⟨title⟩}
```

This is used when the indexcounter option creates records with the wrglossary counter. This ensures that the page name is shown rather than the value of the wrglossary counter.

If the corresponding \glsxtr⟨*counter*⟩locfmt hasn't been defined, \glsxtrdisplay-locnameref will do one of the following:

- if ⟨*title*⟩ is empty or the counter is page, \glsxtrnamereflink with the title set to the location;
- otherwise it will do:

```
\glsxtrtitlednamereflink{⟨format⟩}{⟨location⟩}{⟨title⟩}
{⟨file⟩}
```

which uses \glsxtrnamereflink with the given ⟨*title*⟩ and \glsxtrrecent-anchor as the anchor. This has a better chance of matching the title with the anchor, but it's not guaranteed as some anchors are created without a title. This is defined as:

```
\newcommand{\glsxtrtitlednamereflink}[4]{%
 \glsxtrnamereflink{#1}{#2}{\glsxtrrecentanchor}
{#4}%
}
```

This shows the formatted title with the recent anchor. The location isn't shown. If you would prefer to just show the location and use the anchor corresponding to the location counter:

```
\renewcommand{\glsxtrtitlednamereflink}[4]{%
 \glsxtrnamereflink{#1}{#2}{\glsxtrlocation-
anchor}{#3}%
}
```

If ⟨*file*⟩ is empty an internal link is created with:

```
\glsxtrfmtinternalnameref{⟨target⟩}{⟨format⟩}{⟨file⟩}
```

otherwise an external link is created with:

```
\glsxtrfmtexternalnameref{⟨target⟩}{⟨format⟩}{⟨title⟩}{⟨file⟩}
```

The ⟨*file*⟩ argument is set by bib2gls for supplemental locations (obtained from supplemental-locations).

The link is encapsulated with the text-block command whose name is given by ⟨*format*⟩, but `\glshypernumber` is first locally redefined to `\@firstofone` to prevent a conflict with the usual location hyperlink formation. This means that if the ⟨*format*⟩ is `\hyperbf` then it will simply behave like `\textbf`.

The following command is provided but not used by default:

`\glsxtrnameloclink{`⟨*prefix*⟩`}{`⟨*counter*⟩`}{`⟨*format*⟩`}{`⟨*location*⟩`}{`⟨*text*⟩`}{`⟨*file*⟩`}`

This creates a hyperlink to the target in the given external file obtained from the ⟨*prefix*⟩, ⟨*counter*⟩ and ⟨*location*⟩ but uses ⟨*text*⟩ as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

For compactness, `bib2gls` merges normal records if the ⟨*prefix*⟩, ⟨*counter*⟩ and ⟨*location*⟩ all match. (An order of precedence can be provided for format conflicts.) With `nameref` records, you can use the `--merge-nameref-on` switch provided by `bib2gls` v1.8+ to determine how to merge `nameref` records. This switch must be followed by one of the following keywords: `hcounter` (merge on ⟨*hcounter*⟩, default) `href` (merge on ⟨*href*⟩), `title` (merge on ⟨*title*⟩) and `location` (merge on ⟨*location*⟩, as regular records). In all cases, the ⟨*counter*⟩ must also match.

11.6.7. Dual Entry Commands

Dual entries can be defined with entry types like `@dualentry` or `@dualindexabbreviation`. A single entry definition within the `bib` file creates two dependent entries that may be referenced within the document and assigned to different glossaries. The default `selection={recorded and deps}` will ensure that dependent entries are selected, even if they don't have any records. For example, the following `bib` entry:

```
@dualindexsymbol{L,
  name={Lagrangian function},
  symbol={\ensuremath{L}},
  description={a function of generalized co-
ordinates}
}
```

is essentially like:

```
@index{L,
  name={Lagrangian function},
  symbol={\ensuremath{L}},
```

```
  description={a function of generalized co-
ordinates}
}
@symbol{dual.L,
  name={\ensuremath{L}},
  symbol={Lagrangian function},
  description={a function of generalized co-
ordinates}
}
```

but additionally the two entries (the primary `L` and the dual `dual.L`) are dependent. The following document only references the primary entry, but the dependency ensures that that the dual is also selected:

```
\usepackage[record,symbols,nostyles,
 stylemods={tree,bookindex}]{glossaries-extra}
\GlsXtrLoadResources[src={entries},dual-type=
symbols]
\glsdefpostlink{index}{\glsxtrpostlinkAddDescOn-
FirstUse}
\glsdefpostname{index}{%
 \ifglshassymbol{\glscurrententrylabel}
 { (\glsentrysymbol{\glscurrententrylabel})}{}}
\begin{document}
Primary: \gls{L}.
\printunsrtglossary[type=symbols,style=tree]
\printunsrtglossary[title=Index,style=bookindex]
\end{document}
```

The dependency ensures that the dual entry `dual.L` is selected, and the `dual-type` setting adds the dual entry to the `symbols` glossary (which was defined with the `symbols` package option).

It may be useful to have a hyperlink from the entry in one glossary to its dependent in the other glossary. This can be achieved by instructing `bib2gls` to save the label of the entry's opposite using `dual-field`. The value of this resource option indicates the field in which to save the label. If omitted, `dual` is assumed.

The glossary style can then be adapted to check if the field has been set and, if so, to create a hyperlink. The following command is provided to assist with this:

```
\GlsXtrDualBackLink{⟨text⟩}{⟨entry-label⟩}
```

This is defined as:

```
\newcommand*{\GlsXtrDualBackLink}[2]{%
  \glsxtrifhasfield{\GlsXtrDualField}{#2}%
  {\glshyperlink[#1]{\glscurrentfieldvalue}}%
  {#1}%
}
```

This obtains the field from:

```
\GlsXtrDualField                                    initial: dual
```

This expands to `dual`, which is the default for `dual-field` if no value is supplied.

The above example can be adapted as follows:

```
\usepackage[colorlinks]{hyperref}
\usepackage[record,symbols,nostyles,stylemods=
{tree,bookindex}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}, dual-type=
symbols,dual-field]
\glsdefpostlink{index}{\glsxtrpostlinkAddSymbolOn-
FirstUse}
\glsdefpostname{index}{%
 \ifglshassymbol{\glscurrententrylabel}
 { (\GlsXtrDualBackLink{\glsentrysymbol{\glscurrent-
entrylabel}}{\glscurrententrylabel})}{}}
\begin{document}
Primary: \gls{L}.
\printunsrtglossary[type=symbols,style=tree]
\printunsrtglossary[title=Index,style=bookindex]
\end{document}
```

Example 163: Using bib2gls: dual backlinks

Primary: Lagrangian function (L).

Symbols

L (Lagrangian function) a function of generalized co-ordinates

Index

Lagrangian function (L), 1

This checks if the symbol key has been set in both the category post-link hook and the category post-name hook to append the symbol in parentheses. In addition, the category post-name hook adds a link to the corresponding dual entry.

If you prefer instead to have the backlink on the name in both glossaries, then this can more easily be achieved with a resource option such as dual-backlink.

The commands described below were designed for use with bib2gls's dual entries, but may also be used in other contexts where a label may potentially have a number of possible prefixes.

It's possible to use commands like \glsxtrnewgls to create \gls-like commands that automatically insert a label prefix (such as dual. for dual entries). The commands described in this section provide a similar set of \gls-like commands that iterate over a set of possible prefixes until a match is found.

Each possible prefix (which may be empty) is identified by:

```
\glsxtraddlabelprefix{⟨label-prefix⟩}
```

{prefix} These should be listed in order of precedence.

You can prepend a prefix to the list using:

```
\glsxtrprependlabelprefix{⟨label-prefix⟩}
```

which gives it the highest order of precedence.

The list of known prefixes can be (locally) cleared with:

```
\glsxtrclearlabelprefixes
```

You can test if a prefix is already in the list with:

\glsxtrifinlabelprefixlist{⟨label-prefix⟩}{⟨true⟩}{⟨false⟩}

This does ⟨*true*⟩ if the given prefix has been added to the list. No expansion is performed on ⟨*label-prefix*⟩.

In the event that there's no match for any of the prefixes (which will occur on the first LaTeX run before the `glstex` file has been created), the fallback is determined by the conditional:

\ifGlsXtrPrefixLabelFallbackLast ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iftrue

If true, this will fallback on the last prefix, otherwise it will fallback on the first. This conditional can be set to true with:

```
\GlsXtrPrefixLabelFallbackLasttrue
```

and to false with:

```
\GlsXtrPrefixLabelFallbackLastfalse
```

The default is true.

As from v1.49, all possible matches will be recorded using a special syntax if none are found. This ensures there will be at least one match on the first run. However, note that this requires `bib2gls` v3.0+.

In general it's best to avoid adding multiple instances of the same prefix, so you can check with this command before adding a prefix to the list. However, it can be useful to repeat a prefix at the start or end of the list so that it can be used as a fallback for entries that haven't yet been defined.

With the list of possible prefixes set up (including an empty prefix if necessary), you can use:

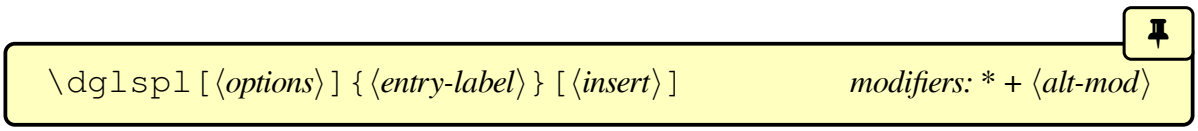
\dgls[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

which behaves like

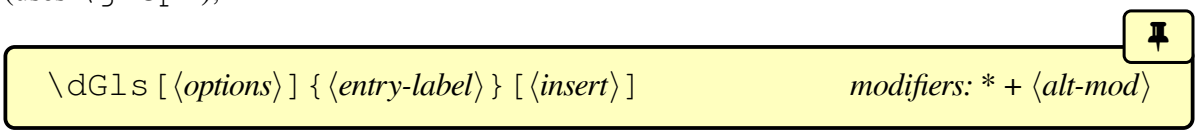
```
\gls[⟨options⟩]{⟨prefix⟩}{entry-label}}[⟨insert⟩]
```

where ⟨*prefix*⟩ is the first prefix in the list such that ⟨*prefix*⟩⟨*label*⟩ matches a defined entry. This requires `bib2gls` v3.0+ to work properly on the first LaTeX run (when no entries are defined).

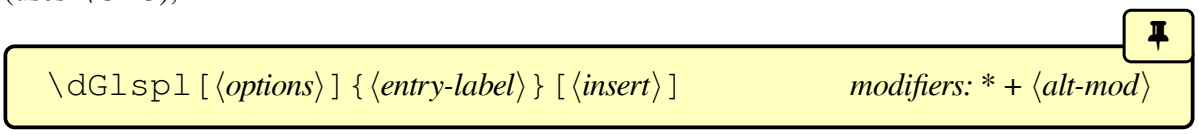
There are also analogous commands for the plural and case-changing versions:



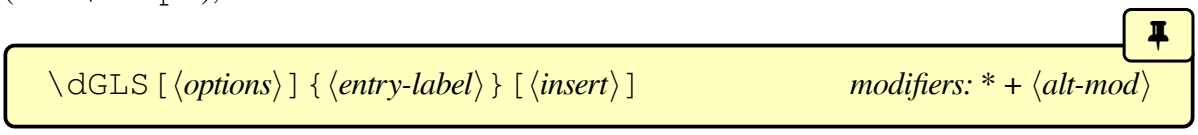
(uses \glspl),



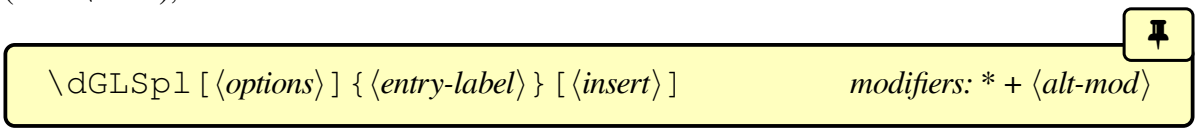
(uses \Gls),



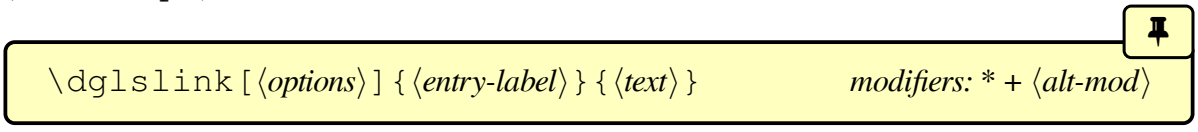
(uses \Glspl),



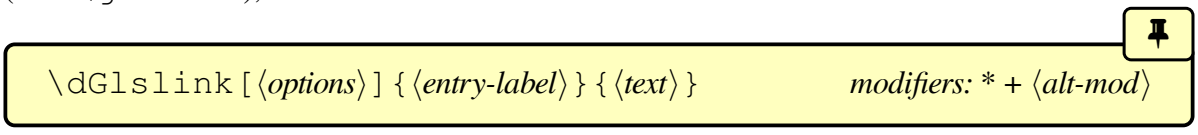
(uses \GLS),



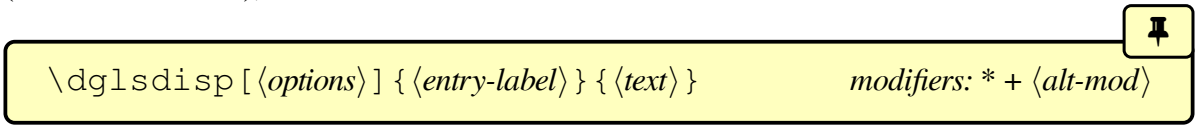
(uses \GLSpl),



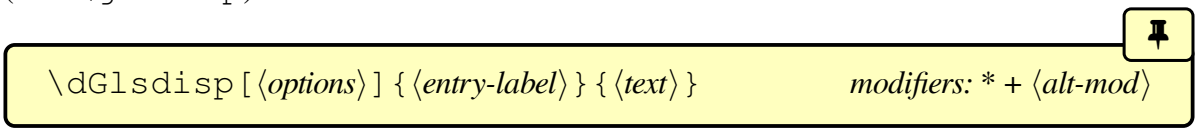
(uses \glslink),



(uses \Glslink),



(uses \glsdisp) and



(uses \Glsdisp).

If you want to use a specific field you can instead use:

\dglsfield[⟨*options*⟩]{⟨*entry-label*⟩}{⟨*field-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

This will find the first match that has the given field set (that is, the field value has been defined and is not empty or \relax). The field should be identified by its internal field label. There are also case-changing versions:

\dGlsfield[⟨*options*⟩]{⟨*entry-label*⟩}{⟨*field-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

which applies a sentence case change and

\dGLSfield[⟨*options*⟩]{⟨*entry-label*⟩}{⟨*field-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

which applies all caps. Note that at least one of the potential labels must have the given field set in order for the reference to be correctly resolved. For example:

```
\dglsfield{pi}{symbol}
```

If you find this a bit long-winded to type and you want to provide a shorter command that recognises the modifiers, then you can use:

\newdglsfield[⟨*default-options*⟩]{⟨*field*⟩}{⟨*cs*⟩}

For example:

```
\newdglsfield{symbol}{\sym}
\sym{pi}
```

If you also want sentence case and all caps versions use:

\newdglsfieldlike[⟨*default-options*⟩]{⟨*field*⟩}{⟨*cs*⟩}{⟨*Cs*⟩}{⟨*CS*⟩}

where ⟨*cs*⟩ is the command without a case-change (which will use \dglsfield), ⟨*Cs*⟩ is the sentence case command (which will use \dGlsfield) and ⟨*CS*⟩ is the all caps command (which will use \dGLSfield). This will also use \glsmfuaddmap and \glsmfu-blocker to establish the sentence case mapping from ⟨*cs*⟩ to ⟨*Cs*⟩ and block the case change for ⟨*CS*⟩.

No expansion is performed on ⟨*default-options*⟩ or ⟨*field*⟩ while the new commands are being defined.

Information is written to the transcript file with `\GlossariesExtraInfo` if a fallback is considered. Note that this shouldn't be considered a warning as the fallback might be desired, but if the wrong output is produced this information may explain the selection.

With commands like `\gls`, it's easy to create a shortcut command that can add some content that varies according to the entry. For example, suppose you want to keep displaying both the entry's symbol followed by the entry's usual text but you don't want to alter the post-link hook as you don't always want to do this. You could, for example define:

```
\newcommand\symgls[1]{\glssymbol{#1}\gls{#1}}
```

This can be harder to do with commands like `\dgls` as you may need to access the actual label. For example, suppose instead of simply showing a field (the symbol in the above example), you need to access the category. A definition like the above `\symgls` is no longer simple as the argument passed is no longer the actual label.

To help with this situation, the `\dgls`-like commands all use a hook after the label is found and before the actual corresponding `\gls`-like action is performed:

```
\predglshook{⟨entry-label⟩}
```

The argument is the actual found label. Note that if a label isn't found (which will occur on the first LaTeX run), this hook won't be used.

There is a similar hook for `\glslink` and `\glsdisp`:

```
\predglslinkhook{⟨entry-label⟩}
```

And likewise for the `\dglsfield` set of commands:

```
\predglsfieldhook{⟨entry-label⟩}
```

The family of `\dglsfield` commands all define:

```
\dglsfieldcurrentfieldlabel
```

This expands to the given field label.

If the requested field isn't set (according to \ifcsvoid) then the fallback field will be tried instead.

\dglsfieldfallbackfieldlabel *initial:* text

This expands to the fallback field, which defaults to the text field.

If you need to know whether the requested field or the fallback field was used, the following will be set to the actual field used.

\dglsfieldactualfieldlabel

For example, suppose the file entries.bib contains:

```
@index{duck}
@dualindexabbreviation{svm,
  short={SVM},
  long={support vector machine}
}
@dualindexsymbol{pi,
  symbol={\ensuremath{\pi}},
  description=ratio of a circle's circumference to its diameter
}
```

and suppose the document code is:

```
\documentclass{article}
\usepackage{hyperref}
\usepackage[record,abbreviations,symbols]
{glossaries-extra}

\newcommand{\bibglsnewdualindexsymbolsecondary}[5]
{%
 \longnewglossaryentry*{#1}{name={#3},category=
{symbol},
   type={symbols},symbol={#4},#2}{#5}%
}

\GlsXtrLoadResources[src={entries}]
```

```
\begin{document}
First use: \gls{duck}, \gls{svm}, \gls{pi}.
Next use: \gls{duck}, \gls{svm}, \gls{pi}.
\printunsrtglossaries
\end{document}
```

This uses the default empty primary prefix and `dual.` for the dual prefix, so `\gls{svm}` is referencing the primary entry, which is (essentially) an `@index` type not an abbreviation. It therefore doesn't follow the abbreviation style, and it also hyperlinks to the index not to the list of abbreviations. Similarly for `\gls{pi}`, which references the primary `@index` entry rather than the symbol.

What's really needed is:

```
\gls{duck}, \gls{dual.svm}, \gls{dual.pi}.
```

or use `label-prefix` and `dual-prefix` to set the label prefixes:

```
\GlsXtrLoadResources[src={entries},
 label-prefix={idx.},dual-prefix={}
]
```

then only the entries without a dual need a prefix:

```
\gls{idx.duck}, \gls{svm}, \gls{pi}.
```

Using `\glsxtrnewgls` or `\glsxtrnewglslike` (see §5.7) to define the custom `\idx`:

```
\glsxtrnewgls{idx.}{\idx}
```

the entry references can be simplified to:

```
\idx{duck}, \glssvm, \glspi.
```

but this requires remembering which terms have duals.

An alternative is to use `\dgls` instead:

```
\GlsXtrLoadResources[src={entries},
 combine-dual-locations=primary]

\glsxtraddlabelprefix{dual.}
\glsxtraddlabelprefix{}

\begin{document}
First use: \dgls{duck}, \dgls{svm}, \dgls{pi}.
Next use: \dgls{duck}, \dgls{svm}, \dgls{pi}.
\printunsrtglossaries
\end{document}
```

Example 164: Using bib2gls: dual entry label prefixes

First use: duck, support vector machine (SVM), π. Next use: duck, SVM, π.

Glossary

duck 1

pi ratio of a circle's circumference to its diameter 1

SVM 1

Symbols

π ratio of a circle's circumference to its diameter

Abbreviations

SVM support vector machine

On the first LaTeX call (when the glstex file doesn't exist), neither dual.svm nor svm exists, so \dgls uses the last prefix (which is empty in this case). This means that on the first LaTeX run, \dgls{svm} behaves like \gls{svm}, which adds a record for the primary svm entry. The default primary-dual dependency means that this will cause both the primary (svm) and dual (dual.svm) entry to be selected. The location will be added to the

primary entry's location list, unless overridden by resource options, such as combine-dual-locations.

Once bib2gls has been run and the glstex file exists, then dual.svm exists. So \dgls{svm} will again first try dual.svm (as dual. is the first in the list of label prefixes). That now exists, so \dgls{svm} now behaves like \gls{dual.svm}, which follows the abbreviation style and hyperlinks to the list of abbreviations. Similarly for the index-symbol combination dual.pi and pi.

In the case of \dgls{duck}, which doesn't have a dual, the label dual.duck never exists, so that's never selected. However, when there's no match, such as when the glstex file doesn't exist, the duck entry will be recorded with both the dual. prefix and the empty prefix. This allows bib2gls to test which prefix+label combination matches.

If you haven't used combine-dual-locations an extra bib2gls+LaTeX run may be required to correct the location lists.

If you change the label prefixes, remember to update the corresponding \glsxtraddlabelprefix{⟨*label-prefix*⟩}. If no prefixes have been added to the list (or if the list is cleared), just an empty prefix is assumed.

Note that bib2gls v1.8+ provides hooks that identify the label prefixes in the glstex file:

```
\bibglstertiaryprefixlabel{⟨label-prefix⟩}
\bibglsdualprefixlabel{⟨label-prefix⟩}
\bibglsprimaryprefixlabel{⟨label-prefix⟩}
```

These do nothing by default, but they can be defined before the resource file is loaded to set up the prefix list. For example:

```
\newcommand{\bibglsprimaryprefixlabel}[1]{%
 \glsxtraddlabelprefix{#1}}
\newcommand{\bibglsdualprefixlabel}[1]{%
 \glsxtrprependlabelprefix{#1}}
\GlsXtrLoadResources[src={entries}]
```

Remember that this will only have an effect once the glstex file has been created. The prefix list will be empty on the first run (which is treated as a single empty prefix). If this isn't a suitable fallback, it may be necessary to add one after all the resource commands:

```
\newcommand{\bibglsprimaryprefixlabel}[1]{%
 \glsxtraddlabelprefix{#1}}
\newcommand{\bibglsdualprefixlabel}[1]{%
 \glsxtrprependlabelprefix{#1}}
\GlsXtrLoadResources[src={entries},label-prefix=
{idx.}]
\glsxtraddlabelprefix{idx.}
```

Although this rather defeats the purpose of using the hooks as you still have to keep track of the fallback prefix.

11.6.8. Supplementary Commands

```
\glsdisplaynumberlist{⟨entry-label⟩}
```

This command is provided by the base glossaries package, but it requires hooking into the location list formatting within the glossary to obtain the list. This information is more easily available with bib2gls, which stores the formatted location list in the location field, so glossaries-extra-bib2gls redefines this command to use that field value. Likewise for:

```
\glsentrynumberlist{⟨entry-label⟩}
```

which is redefined to simply do:

```
\glsxtrusefield{#1}{location}
```

```
\IfTeXParserLib{⟨TEX parser lib code⟩}{⟨LATEX code⟩}
```

This command is defined by glossaries-extra-bib2gls to expand to LaTeX code, but is defined by the TeX parser library used by bib2gls's interpreter to expand to TeX parser lib code. May be used in the @preamble or in field values to provide bib2gls with alternative content.

There is a similar command with a reversed order of arguments:

```
\IfNotBibGls{⟨LATEX code⟩}{⟨bib2gls code⟩}
```

This command is defined by glossaries-extra-bib2gls to expand to LaTeX code but is defined specifically by bib2gls's interpreter to expand to ⟨*bib2gls*⟩ code. If the TeX parser library is

used by another application (such as the conversion tools provided with bib2gls), the command will be defined to match glossaries-extra-bib2gls.

\glsxtrprovidecommand{⟨cs⟩}[⟨n⟩][⟨default⟩]{⟨definition⟩} *modifier: **

This command is intended for use in @preamble. It's simply defined to \providecommand in glossaries-extra-bib2gls but bib2gls's interpreter treats it as \renewcommand. This means that you can override bib2gls's internal definition of a command without overriding the command definition in the document (if it's already defined before the resource file is input). For example:

```
@preamble{"\glsxtrprovidecommand{\int}{integral}"}
```

This will force bib2gls to treat \int as the word "integral" to assist sorting but if this preamble code is written to the glstex file (as it is by default) then it won't override the current definition (provided by the kernel or redefined by a package).

The helper commands in the resource files are defined using \providecommand. For many of them, if you want to provide an alternative definition then you need to define the command before the resource file is loaded. There are a few that may be redefined afterwards but if you use \renewcommand then you will get an error on the first LaTeX run when the glstex file doesn't exist. In this case, you may prefer to use:

\glsrenewcommand{⟨cs⟩}[⟨n⟩][⟨default⟩]{⟨definition⟩} *modifier: **

This behaves like \renewcommand but only generates a warning rather than an error if the command isn't already defined so it won't interrupt the document build.

\GlsXtrIndexCounterLink{⟨text⟩}{⟨entry-label⟩}

For use with the indexcounter package option and the save-index-counter resource option. This command creates a hyperlink (if supported) to the target obtained from the indexcounter field (which will be set in the glstex file, if applicable).

If then indexcounter field is set for the entry given by ⟨*entry-label*⟩, this command does \hyperref[wrglossary.⟨value⟩]{⟨text⟩}, where ⟨*value*⟩ is the value of the indexcounter field. If the field isn't set or if \hyperref hasn't been defined, this just does ⟨*text*⟩. See the bib2gls manual (v1.4+) for further details.

The glossaries-extra-bib2gls package also provides definitions of the missing mathematical Greek commands: \Alpha, \Beta, \Epsilon, \Zeta, \Eta, \Iota, \Kappa, \Mu, \Nu, \Omicron, \Rho, \Tau, \Chi, \Digamma, \omicron. These are all defined with \providecommand, so they won't override any definitions provided by any

package loaded before glossaries-extra. Since bib2gls's interpreter recognises these commands, using them instead of explicitly using the Latin characters with the same shape helps to keep the Greek symbols together when sorting. Similarly, if upgreek has been loaded, the missing upright Greek commands are also provided: \Upalpha, \Upbeta, \Upepsilon, \Upzeta, \Upeta, \Upiota, \Upkappa, \Upmu, \Upnu, \Upomicron, \Uprho, \Uptau, \Upchi, \upomicron.

12. Auto-Indexing

It's possible that you may also want a normal index as well as the glossary, and you may want entries to automatically be added to the index (as in this document). There are two attributes that govern this: `indexname` and `dualindex`.

The auto-indexing is designed for `makeindex` syntax. If you've used the `xindy` package option, the automatic escaping of `xindy` special characters in the `sort` field may result in an incorrect sort value for the `\index` command used by the auto-indexing. Note also that `texindy` has a fixed set of special characters (corresponding to `makeindex`'s defaults) that can't be customized. You may want to consider using `bib2gls` and its dual entries as an alternative approach.

The `\glsxtrpostnamehook` macro, used at the end of `\glossentryname`, `\Glossentryname`, `\GLOSSentryname` and `\GlossEntryName`, checks the `indexname` attribute for the category associated with that entry. Since `\glossentryname` is used in the default glossary styles, this makes a convenient way of automatically indexing each entry name at its location in the glossary without fiddling around with the value of the `name` key.

The internal macro used by the glossaries package to write the information to the external glossary file is modified to check for the `dualindex` attribute.

In both cases, the indexing is done through:

`\glsxtrdoautoindexname{`⟨*entry-label*⟩`}{`⟨*attribute*⟩`}`

This uses the standard `\index` command with the sort value taken from the entry's `sort` key and the actual value set to `\glsentryname{`⟨*entry-label*⟩`}`. There are user-level commands available to change the sort and actual value used by the automated index.

The actual value is given by:

`\glsxtrautoindexentry{`⟨*entry-label*⟩`}`

where ⟨*entry-label*⟩ is the entry's label. The default definition is:

```
\newcommand*{\glsxtrautoindexentry}[1]{\string\glsentryname{#1}}
```

Note the use of \string to prevent \glsentryname from being expanded as it's written to the index file.

The sort value is assigned using:

\glsxtrautoindexassignsort{⟨*entry-label*⟩}

where ⟨*entry-label*⟩ is the entry label and ⟨*cs*⟩ is the command which needs to be set to the sort value. The default definition is:

```
\newcommand*{\glsxtrautoindexassignsort}[2]{%
  \glsletentryfield{#1}{#2}{sort}%
}
```

After this macro is called, ⟨*cs*⟩ is then processed to escape any of makeindex's special characters. Note that this escaping is only performed on the sort not on the actual value. The escaping of the sort value is performed by

\glsxtrautoindexesc

You can redefine this to do nothing if you want to omit the escaping. You may want to consider providing another field to obtain the appropriate sort value if the one provided in the sort field isn't suitable (because it may already have had special characters escaped or it may be a numeric value in the case of sort by use or definition).

The command used to perform the actual indexing is:

\glsxtrautoindex *initial:* \index

This just does \index{⟨*text*⟩} by default.

The entry's parent field isn't referenced in this automated indexing.

For example, to index the value of the first key, instead of the name key:

```
\renewcommand*{\glsxtrautoindexentry}[1]
{\string\glsentryfirst{#1}}
```

and if the sort value also needs to be set to the long field, if present, otherwise the sort field:

```
\renewcommand*{\glsxtrautoindexassignsort}[2]{%
  \ifglshaslong{#2}%
  {\glsletentryfield{#1}{#2}{long}}%
  {\glsletentryfield{#1}{#2}{sort}}%
}
```

If the value of the attribute is "true", no encap will be added, otherwise the encap will be the attribute value. For example:

```
\glssetcategoryattribute{general}{indexname}{textbf}
```

will set the encap to `textbf` which will display the relevant page number in bold whereas

```
\glssetcategoryattribute{general}{dualindex}{true}
```

won't apply any formatting to the page number in the index.

The location used in the index will always be the page number not the counter used in the glossary. (Unless some other loaded package has modified the definition of `\index` to use some thing else.)

By default the `format` key won't be used with the `dualindex` attribute. You can allow the `format` key to override the attribute value by using the preamble-only command:

```
\GlsXtrEnableIndexFormatOverride
```

If you use this command and hyperref has been loaded, then the theindex environment will be modified to redefine `\glshypernumber` to allow formats that use that command.

The `dualindex` attribute will still be used on subsequent use even if the `indexonlyfirst` attribute (or `indexonlyfirst` package option) is set. However, the `dualindex` attribute will honour the `noindex` key.

The `\glsxtrdoautoindexname` command will attempt to escape any of `makeindex`'s special characters, but there may be special cases where it fails, so take care. This assumes the default `makeindex` actual, level, quote and encap values (unless any of the commands `\actualchar`, `\levelchar`, `\quotechar` or `\encapchar` have been defined before glossaries-extra is loaded).

If this isn't the case, you can use the following preamble-only commands to set the correct characters.

Be very careful of possible shifting category codes!

`\GlsXtrSetActualChar{`⟨*character*⟩`}`

Set the actual character to ⟨*char*⟩.

`\GlsXtrSetLevelChar{`⟨*character*⟩`}`

Set the level character to ⟨*char*⟩.

`\GlsXtrSetEscChar{`⟨*character*⟩`}`

Set the escape (quote) character to ⟨*char*⟩.

`\GlsXtrSetEncapChar{`⟨*character*⟩`}`

Set the encap character to ⟨*char*⟩.

13. On-the-Fly Document Definitions

The commands described here may superficially look like ⟨*word*⟩`\index{`⟨*word*⟩`}`, but they behave rather differently. If you want to use `\index` then just use `\index`.

The base glossaries package advises against defining entries in the document environment. As mentioned in §2.4, this ability is disabled by default with glossaries−extra but can be enabled using the `docdef` package options.

Although this can be problematic, the glossaries−extra package provides a way of defining and using entries within the document environment without the tricks used with the `docdef` option. *There are limitations with this approach, so take care with it.* This function is disabled by default, but can be enabled using the preamble-only command:

`\GlsXtrEnableOnTheFly` *modifier:* *

When used, this defines the commands described below. The starred version was provided to help support UTF-8 with inputenc. Recent versions of the LaTeX kernel now provide better support. Both glossaries and glossaries−extra have been updated to help integrate the new kernel features, so the unstarred version may also work with UTF-8.

If you use the starred version of `\GlsXtrEnableOnTheFly` don't use any commands in the ⟨*label*⟩, even if they expand to just text.

`\glsxtr[`⟨*gls-options*⟩`][`⟨*dfn-options*⟩`]{`⟨*entry-label*⟩`}`

If an entry with the label ⟨*entry-label*⟩ has already been defined, this just does `\gls[`⟨*gls-options*⟩`]{`⟨*entry-label*⟩`}`. If ⟨*entry-label*⟩ hasn't been defined, this will define the entry using:

```
\newglossaryentry{⟨entry-label⟩}{name={entry-label},
 category={\glsxtrcat},
 description={\nopostdesc},
 ⟨dfn-options⟩}
```

The ⟨*label*⟩ must contain any non-expandable commands, such as formatting commands or problematic characters. If the term requires any of these, they must be omitted from the ⟨*entry-label*⟩ and placed in the `name` key within the optional argument ⟨*dfn-options*⟩.

The second optional argument ⟨*dfn-options*⟩ should be empty if the entry has already been defined, since it's too late for them. If it's not empty, a warning will be generated with:

```
\GlsXtrWarning{⟨options⟩}{⟨entry⟩}
```

For example, this warning will be generated on the second instance of `\glsxtr` below:

```
\glsxtr[][pluralgeese]{goose}
% ... later
\glsxtr[][pluralgeese]{goose}
```

If you are considering doing something like:

```
\newcommand*{\goose}{\glsxtr[][plural={geese}]
{goose}}
\renewcommand*{\GlsXtrWarning}[2]{}
% ... later
\goose\ some more text here
```

then don't bother. It's simpler and less problematic to just define the entries in the preamble with `\newglossaryentry` and then use `\gls` in the document.

There are also plural and case-changing alternatives to `\glsxtr`.

```
\glsxtrpl[⟨gls-options⟩][⟨dfn-options⟩]{⟨entry-label⟩}
```

This is like `\glsxtr` but uses `\glspl` instead of `\gls`.

```
\Glsxtr[⟨gls-options⟩][⟨dfn-options⟩]{⟨entry-label⟩}
```

This is like `\glsxtr` but uses `\Gls` instead of `\gls`.

```
\Glsxtrpl[⟨gls-options⟩][⟨dfn-options⟩]{⟨entry-label⟩}
```

This is like `\glsxtr` but uses `\Glspl` instead of `\gls`.

The category is set to:

\glsxtrcat *initial:* general

This should simply expand to the required category label. The default definition is general.

The commands \glsxtr, \glsxtrpl, \Glsxtr and \Glsxtrpl can't be used after the glossaries have been displayed (through \printglossary etc). It's best not to mix these commands with the standard glossary commands, such as \gls or there may be unexpected results.

14. Supplementary Files

The glossaries−extra package comes with some additional files. Those listed in §14.1 provide dummy entries for testing various styles. They should be placed on TEX's path.

There are also some sample files listed in §14.2. These should be located in the package documentation directory.

14.1. Dummy Files for Testing

The base glossaries package provides files with dummy entries for testing. The glossaries−extra package provides an additional file with entries.

example-glossaries-xr.tex

This file contains entries that have the see, seealso or alias keys set.
There are also bib files corresponding to all the available tex files for use with bib2gls.

example-glossaries-acronym.bib

Corresponds to example-glossaries-acronym.tex

example-glossaries-acronym-desc.bib

Corresponds to example-glossaries-acronym-desc.tex

example-glossaries-acronyms-lang.bib

Corresponds to example-glossaries-acronyms-lang.tex

example-glossaries-brief.bib

Corresponds to example-glossaries-brief.tex

example-glossaries-childmultipar.bib

Corresponds to example-glossaries-childmultipar.tex

example-glossaries-childnoname.bib

Corresponds to example-glossaries-childnoname.tex

example-glossaries-cite.bib

Corresponds to example-glossaries-cite.tex

example-glossaries-constants.bib

Corresponds to example-glossaries-constants.tex

example-glossaries-images.bib

Corresponds to example-glossaries-images.tex

example-glossaries-long.bib

Corresponds to example-glossaries-long.tex

example-glossaries-longchild.bib

Corresponds to example-glossaries-longchild.tex

example-glossaries-multipar.bib

Corresponds to example-glossaries-multipar.tex

example-glossaries-parent.bib

Corresponds to example-glossaries-parent.tex

example-glossaries-symbolnames.bib

Corresponds to example-glossaries-symbolnames.tex

example-glossaries-symbols.bib

Corresponds to example-glossaries-symbols.tex

example-glossaries-url.bib

Corresponds to example-glossaries-url.tex

example-glossaries-user.bib

Corresponds to example-glossaries-user.tex

example-glossaries-utf8.bib

Corresponds to example-glossaries-utf8.tex

example-glossaries-xr.bib

Corresponds to example-glossaries-xr.tex

14.2. Sample Files

The glossaries-extra package comes with some sample files that are listed below. There are also sample files provided with the glossaries package and with bib2gls. See also the Dickimaw Books Gallery.[1]

sample.tex

Simple sample file that uses one of the dummy files provided by the glossaries package for testing.

[1] dickimaw-books.com/gallery

```
pdflatex sample
makeglossaries sample
pdflatex sample
pdflatex sample
```

sample-abbr-styles.tex

```
pdflatex sample-abbr-styles
makeglossaries sample-abbr-styles
pdflatex sample-abbr-styles
pdflatex sample-abbr-styles
```

Demonstrates all predefined abbreviation styles.

sample-mixture.tex

```
pdflatex sample-mixture
makeglossaries sample-mixture
pdflatex sample-mixture
makeglossaries sample-mixture
pdflatex sample-mixture
```

General entries, acronyms and initialisms all treated differently.

sample-name-font.tex

```
pdflatex sample-name-font
makeglossaries sample-name-font
pdflatex sample-name-font
pdflatex sample-name-font
```

Categories and attributes are used to customize the way different entries appear.

sample-abbrv.tex

```
pdflatex sample-abbrv
makeglossaries sample-abbrv
pdflatex sample-abbrv
pdflatex sample-abbrv
```

General abbreviations.

sample-acronym.tex

```
pdflatex sample-acronym
makeglossaries sample-acronym
pdflatex sample-acronym
```

Acronyms aren't initialisms and don't expand on first use.

sample-acronym-desc.tex

```
pdflatex sample-acronym-desc
makeglossaries sample-acronym-desc
pdflatex sample-acronym-desc
makeglossaries sample-acronym-desc
pdflatex sample-acronym-desc
```

Acronyms that have a separate long form and description.

sample-crossref.tex

```
pdflatex sample-crossref
makeglossaries sample-crossref
pdflatex sample-crossref
```

Unused entries that have been cross-referenced automatically are added at the end of the document.

sample-indexhook.tex

```
pdflatex sample-indexhook
makeglossaries sample-indexhook
pdflatex sample-indexhook
```

Use the index hook to track which entries have been indexed (and therefore find out which ones haven't been indexed).

sample-footnote.tex

```
pdflatex sample-footnote
makeglossaries sample-footnote
pdflatex sample-footnote
```

Footnote abbreviation style that moves the footnote marker outside of the hyperlink generated by `\gls` and moves it after certain punctuation characters for neatness.

sample-undef.tex

```
pdflatex sample-undef
makeglossaries sample-undef
pdflatex sample-undef
pdflatex sample-undef
```

Warn on undefined entries instead of generating an error.

sample-mixed-abbrv-styles.tex

```
pdflatex sample-mixed-abbrv-styles
makeglossaries sample-mixed-abbrv-styles
pdflatex sample-mixed-abbrv-styles
```

Different abbreviation styles for different entries.

sample-initialisms.tex

```
pdflatex sample-initialisms
makeglossaries sample-initialisms
pdflatex sample-initialisms
```

Automatically insert dots into initialisms.

sample-postdot.tex

```
pdflatex sample-postdot
makeglossaries sample-postdot
pdflatex sample-postdot
```

Another initialisms example.

sample-postlink.tex

```
pdflatex sample-postlink
makeglossaries sample-postlink
pdflatex sample-postlink
```

Automatically inserting text after the link text produced by commands like `\gls` (outside of hyperlink, if present).

sample-header.tex

```
pdflatex sample-header
makeglossaries sample-header
pdflatex sample-header
pdflatex sample-header
```

Using entries in section/chapter headings.

sample-autoindex.tex

```
pdflatex sample-autoindex
makeglossaries sample-autoindex
pdflatex sample-autoindex
makeindex sample-autoindex
pdflatex sample-autoindex
```

Using the `dualindex` and `indexname` attributes to automatically add glossary entries to the index (in addition to the glossary location list).

sample-autoindex-hyp.tex

```
pdflatex sample-autoindex-hyp
makeglossaries sample-autoindex-hyp
pdflatex sample-autoindex-hyp
makeindex sample-autoindex-hyp
pdflatex sample-autoindex-hyp
```

As previous but uses hyperref.

sample-nested.tex

```
pdflatex sample-nested
makeglossaries sample-nested
pdflatex sample-nested
```

Using `\gls` within the value of the `name` key.

sample-entrycount.tex

```
pdflatex sample-entrycount
pdflatex sample-entrycount
makeglossaries sample-entrycount
pdflatex sample-entrycount
```

Enable entry-use counting (only index if used more than n times, see §6.1).

sample-unitentrycount.tex

```
pdflatex sample-unitentrycount
pdflatex sample-unitentrycount
makeglossaries sample-unitentrycount
pdflatex sample-unitentrycount
```

Enable use of per-unit entry-use counting (§6.1).

sample-onelink.tex

```
pdflatex sample-onelink
makeglossaries sample-onelink
pdflatex sample-onelink
```

Using the per-unit entry counting (§6.1) to only have one hyperlink per entry per page.

sample-linkcount.tex

```
pdflatex sample-linkcount
makeglossaries sample-linkcount
pdflatex sample-linkcount
```

Using link counting (§6.2) to only have one hyperlink per entry.

sample-pages.tex

```
pdflatex sample-pages
makeglossaries sample-pages
pdflatex sample-pages
pdflatex sample-pages
```

Insert “page” or “pages” before the location list.

sample-altmodifier.tex

```
pdflatex sample-altmodifier
makeglossaries sample-altmodifier
pdflatex sample-altmodifier
pdflatex sample-altmodifier
```

Set the default options for commands like \gls and add an alternative modifier.

sample-mixedsort.tex

```
pdflatex sample-mixedsort
pdflatex sample-mixedsort
```

Uses the optional argument of \makeglossaries to allow a mixture of \printglossary and \printnoidxglossary.

sample-external.tex

```
pdflatex sample-external
makeglossaries sample-external
pdflatex sample-external
```

Uses the targeturl attribute to allow for entries that should link to an external URL rather than to an internal glossary.

sample-fmt.tex

```
pdflatex sample-fmt
makeglossaries sample-fmt
pdflatex sample-fmt
```

Provides text-block commands associated with entries in order to use \glsxtrfmt.

sample-alias.tex

```
pdflatex sample-alias
makeglossaries sample-alias
pdflatex sample-alias
```

Uses the `alias` key (see §3.4).

sample-alttree.tex

```
pdflatex sample-alttree
makeglossaries sample-alttree
pdflatex sample-alttree
```

Uses the glossaries-extra-stylemods package with the alttree style (see §8.6.5).

sample-alttree-sym.tex

```
pdflatex sample-alttree-sym
makeglossaries sample-alttree-sym
pdflatex sample-alttree-sym
```

Another alttree example that measures the symbol widths.

sample-alttree-marginpar.tex

```
pdflatex sample-alttree-marginpar
makeglossaries sample-alttree-marginpar
pdflatex sample-alttree-marginpar
```

Another alttree example that puts the location list in the margin.

sample-onthefly.tex

```
pdflatex sample-onthefly
makeglossaries sample-onthefly
pdflatex sample-onthefly
```

Using on-the-fly commands. Terms with accents must have the `name` key explicitly set.

sample-onthefly-xetex.tex

```
xelatex sample-onthefly-xetex
makeglossaries sample-onthefly-xetex
xelatex sample-onthefly-xetex
```

Using on-the-fly commands with XeLaTeX. Terms with UTF-8 characters don't need to have the `name` key explicitly set. Terms that contain commands must have the `name` key explicitly set with the commands removed from the label.

sample-onthefly-utf8.tex

```
pdflatex sample-onthefly-utf8
makeglossaries sample-onthefly-utf8
pdflatex sample-onthefly-utf8
```

Tries to emulate the previous sample file for use with LaTeX through the starred version of `\GlsXtrEnableOnTheFly`. This is a bit iffy and may not always work. Terms that contain commands must have the `name` key explicitly set with the commands removed from the label.

sample-accsupp.tex

```
pdflatex sample-accsupp
makeglossaries sample-accsupp
pdflatex sample-accsupp
```

Integrate glossaries-accsupp.

sample-prefix.tex

```
pdflatex sample-prefix
makeglossaries sample-prefix
pdflatex sample-prefix
```

Integrate glossaries-prefix.

sample-suppl-main.tex

```
pdflatex sample-suppl
pdflatex sample-suppl-main
makeglossaries sample-suppl-main
pdflatex sample-suppl-main
```

Uses `thevalue` to reference a location in the supplementary file `sample-suppl.tex`.

`sample-suppl-main-hyp.tex`

```
pdflatex sample-suppl-hyp
makeglossaries sample-suppl-hyp
pdflatex sample-suppl-hyp
pdflatex sample-suppl-main-hyp
makeglossaries sample-suppl-main-hyp
pdflatex sample-suppl-main-hyp
```

A more complicated version to the above that uses the hyperref package to reference a location in the supplementary file `sample-suppl-hyp.tex`.

15. Multi-Lingual Support

There's only one command provided by glossaries−extra that you're likely to want to change in your document and that's `\abbreviationsname` if you use the `abbreviations` package option to automatically create the glossary labelled `abbreviations`. If this command doesn't already exist, it will be defined to "Abbreviations" if babel hasn't been loaded, otherwise it will be defined as `\acronymname` (provided by glossaries), which is language-sensitive.

You can redefine `\abbreviationsname` in the usual way. For example:

```
\renewcommand*{\abbreviationsname}
{List of Abbreviations}
```

Or using babel or polyglossia captions hook:

```
\appto\captionsenglish{%
 \renewcommand*{\abbreviationsname}
{List of Abbreviations}%
}
```

Alternatively you can use the `title` key when you print the list of abbreviations. For example:

```
\printabbreviations[title={List of Abbreviations}]
```

or

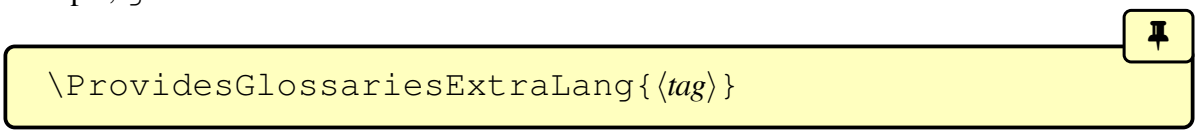
```
\printabbreviations[typeabbreviations,title=
{List of Abbreviations}]
```

The other fixed text commands are the diagnostic messages, which shouldn't appear in the final draft of your document.

The glossaries−extra package has the facility to load language modules (whose filename is in the form `glossariesxtr-`⟨*language*⟩`.ldf`) if they exist, but won't warn if they don't. If glossaries−extra−bib2gls is loaded via the `record` package option then the check for language

resource files will additionally search for an associated language script file given by `glossariesxtr-⟨script⟩.ldf` where ⟨*script*⟩ is the four letter script identifier, such as `Latn`, associated with the given dialect. There's no warning if the associated file isn't found. The script file is loaded after the dialect file.

If you want to write your own language module, you just need to create a file called `glossariesxtr-⟨lang⟩.ldf`, where ⟨*lang*⟩ identifies the language or dialect (see the tracklang package). For example, `glossariesxtr-french.ldf`. The file should start with:



```
\ProvidesGlossariesExtraLang{⟨tag⟩}
```

The simplest code for this file is:

```
\ProvidesGlossariesExtraLang{french}
[2015/12/09 v1.0]

\newcommand*{\glossariesxtrcaptionsfrench}{%
 \def\abbreviationsname{Abr\'eviations}%
}
\glossariesxtrcaptionsfrench

\ifcsdef{captions\CurrentTrackedDialect}
{%
  \csappto{captions\CurrentTrackedDialect}%
  {%
    \glossariesxtrcaptionsfrench
  }%
}%
{%
  \ifcsdef{captions\CurrentTrackedLanguage}
  {%
    \csappto{captions\CurrentTrackedLanguage}%
    {%
      \glossariesxtrcaptionsfrench
    }%
  }%
  {}%
}
```

You can adapt this for other languages by replacing all instances of the language identifier `french` and the translated text `Abr\'eviations` as appropriate. You can also use the `ldf` file to provide rule blocks for a particular language for use with `bib2gls`'s custom sort rule. See §11.6 for further details.

For further information on tracklang, see the tracklang documentation or Localisation with `tracklang.tex`.[a]

[a] `dickimaw-books.com/latex/tracklang/`

This `ldf` file then needs to be put somewhere on TEX's path so that it can be found by glossaries-extra. You might also want to consider uploading it to CTAN[1] so that it can be useful to others. (Please don't send it to me. I already have more packages than I am able to maintain.)

If you additionally want to provide translations for the diagnostic messages used when a glossary is missing, you need to redefine the following commands:

`\GlsXtrNoGlsWarningHead{`⟨*glossary-label*⟩`}{`⟨*file*⟩`}`

This produces the following text in English:

> This document is incomplete. The external file associated with the glossary '⟨*glossary-label*⟩' (which should be called ⟨*file*⟩) hasn't been created.

`\GlsXtrNoGlsWarningEmptyStart`

This produces the following text in English:

> This has probably happened because there are no entries defined in this glossary.

`\GlsXtrNoGlsWarningEmptyMain`

This produces the following text in English:

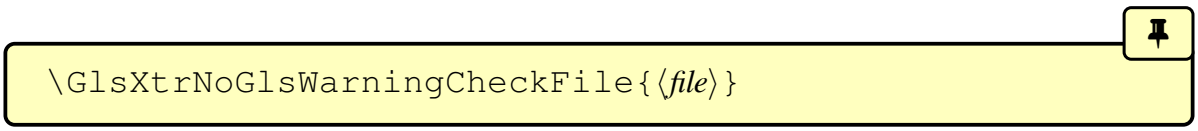
> If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

`\GlsXtrNoGlsWarningEmptyNotMain{`⟨*glossary-label*⟩`}`

This produces the following text in English:

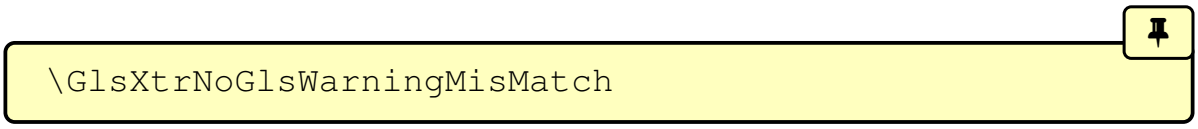
> Did you forget to use `type=`⟨*glossary-label*⟩ when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[`⟨*glossary-label*⟩`]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

[1] `ctan.org/`



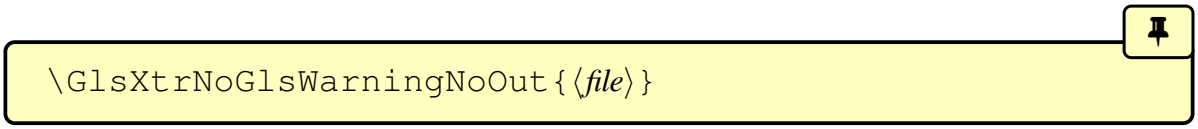
This produces the following text in English:

> Check the contents of the file ⟨*file*⟩. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.



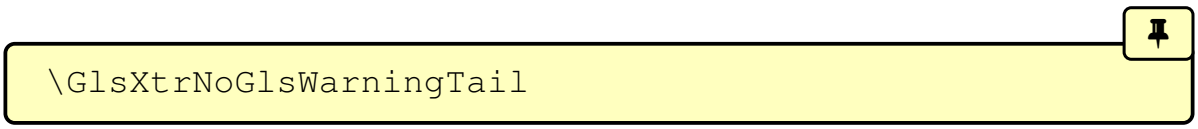
This produces the following text in English:

> You need to either replace `\makenoidxglossaries` with `\makeglossaries` or replace `\printglossary` (or `\printglossaries`) with `\printnoidxglossary` (or `\printnoidxglossaries`) and then rebuild this document.



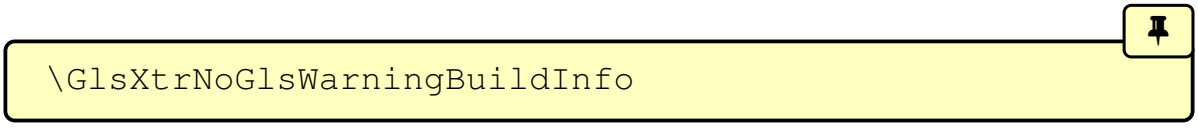
This produces the following text in English:

> The file ⟨*file*⟩ doesn't exist. This most likely means you haven't used `\makeglossaries` or you have used `\nofiles`. If this is just a draft version of the document, you can suppress this message using the `nomissingglstext` package option.

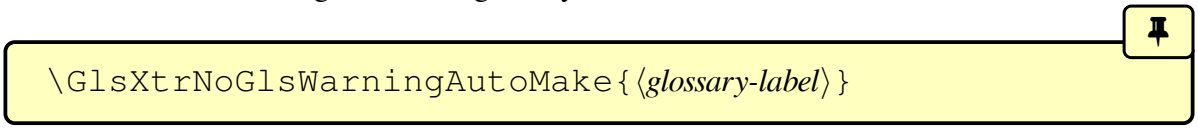


This produces the following text in English:

> This message will be removed once the problem has been fixed.



This is advice on how to generate the glossary files.



This is the message produced when the `automake` option is used, but the document needs a rerun or the shell escape setting doesn't permit the execution of the external application. This command also generates a warning in the transcript file.

See the documented code (`glossaries-extra-code.pdf`) for further details.

Part II.

Summaries and Index

Symbols

Symbol	Description
№	A counter is being described.
	The syntax and usage of a command, environment or option etc.
	A command, environment or option that is now deprecated.
	A command, environment or option that should not be used with glossaries-extra.
	An important message.
	Prominent information.
	LaTeX code to insert into your document.
	The definition of an option value.
	Problematic code which should be avoided.
	How the example code should appear in the PDF.
	An option that takes a value.
>_	A command-line application invocation that needs to be entered into a terminal or command prompt. See also “Incorporating makeglossaries or makeglossaries-lite or bib2gls into the document build[1]”.
	A boolean option that is initially false.
	A boolean option that is initially true.
	Text in a transcript or log file or written to STDOUT or STDERR.
	Code that requires a native Unicode engine (XeLaTeX or LuaLaTeX).
	An option that doesn’t take a value.
	A warning.

[1] dickimaw-books.com/latex/buildglossaries

Terms

American Standard Code for Information Interchange (ASCII)

A single-byte character encoding. Related blog article: Binary Files, Text Files and File Encodings.[1]

Category post-description hook

A post-description hook called `\glsxtrpostdesc`⟨*category*⟩ that is associated with a particular category.

Category post-link hook

A post-link hook called `\glsxtrpostlink`⟨*category*⟩ that is associated with a particular category.

Category post-name hook

A post-name hook called `\glsxtrpostname`⟨*category*⟩ that is associated with a particular category.

Command-line interface (CLI)

An application that doesn't have a graphical user interface. That is, an application that doesn't have any windows, buttons or menus and can be run in a command prompt or terminal.[2]

Display full form

The full form of an abbreviation obtained with the first use of `\gls` or `\glspl` (or case-changing variants). This may or may not produce the same result as the inline full form the corresponding `\glsxtrfull` command, depending on the abbreviation style.

Entry location

The location of the entry in the document (obtained from the location counter or from the `thevalue` option). This defaults to the page number on which the entry has been referenced with any of the `\gls`-like, `\glstext`-like or `\glsadd` commands. An entry may have multiple locations that form a list.

Field

Entry data is stored in fields. These may have a corresponding key used to set the value, such as `name` or `description`, but field values may also be set using commands like `\GlsXtrSetField`, in which case there doesn't need to be a corresponding key. Some fields are

[1] `dickimaw-books.com/blog/binary-files-text-files-and-file-encodings/`
[2] `dickimaw-books.com/latex/novices/html/terminal.html`

considered internal fields. If you are using bib2gls, it will only recognise fields in the bib file that have had a key defined in the document or that are special to bib2gls.

First use

The first time an entry is used by a command that unsets the first use flag (or the first time since the flag was reset). For multi-entry sets, see first use.

First use flag

A conditional that keeps track of whether or not an entry has been referenced by any of the \gls-like commands (which can adjust their behaviour according to whether or not this flag is true). The conditional is true if the entry hasn't been used by one of these commands (or if the flag has been reset) and false if it has been used (or if the flag has been unset). Note that multi-entries have their own flag that's distinct from the first use flags of the individual elements.

First use text

The link text that is displayed on first use of the \gls-like commands.

Group (letters, numbers, symbols)

A logical division within a glossary that is typically a by-product of the indexing application's sorting algorithm. See also Gallery: Logical Glossary Divisions (type vs group vs parent).[3]

Graphical user interface (GUI)

An application that has windows, buttons or menus.

Glossary

Technically a glossary is an alphabetical list of words relating to a particular topic. For the purposes of describing the glossaries and glossaries-extra packages, a glossary is either the list produced by commands like \printglossary or \printunsrtglossary (which may or may not be ordered alphabetically) or a glossary is a set of entry labels where the set is identified by the glossary label or type.

\gls-like

Commands like \gls and \glsdisp that change the first use flag. These commands index the entry (if indexing is enabled), create a hyperlink to the entry's glossary listing (if enabled) and unset the first use flag. These commands end with the post-link hook.

\glstext-like

Commands like \glstext and \glslink that don't change the first use flag. These commands index the entry (if indexing is enabled) and create a hyperlink to the entry's glossary listing (if enabled). These commands end with the post-link hook.

Hierarchical level

A number that indicates how many ancestors an entry has. An entry with no parent has hierarchical level 0. If an entry has a parent then the hierarchical level for the entry is one more than

[3] dickimaw-books.com/gallery/index.php?label=logicaldivisions

the hierarchical level of the parent. Most styles will format an entry according to its hierarchical level, giving prominence to level 0 entries, although some may have a maximum supported limit. The "unsrt" family of commands allow the hierarchical level to be locally adjusted by adding an offset (`leveloffset`) or to have hierarchy ignored (`flatten`).

Ignored glossary

A glossary that has been defined using a command like `\newignoredglossary`. These glossaries are omitted by iterative commands, such as `\printglossaries` and `\printunsrtglossaries`. An ignored glossary can only be displayed with `\printunsrtglossary` or `\printunsrtinnerglossary`.

Ignored location (or record)

A location that uses `glsignore` as the encap. With `bib2gls`, this indicates that the entry needs to be selected but the location isn't added to the location list. With other methods, this will simply create an invisible location, which can result in unwanted commas if the location list has other items.

Indexing application

An application (piece of software) separate from TEX/LATEX that collates and sorts information that has an associated page reference. Generally the information is an index entry but in this case the information is a glossary entry. There are two main indexing applications that are used with TEX: `makeindex` and `xindy`. (There is also a new application called `xindex`, but this isn't supported by glossaries or glossaries−extra.) The glossaries−extra package additionally supports `bib2gls`. These are all CLI applications.

Indexing (or recording)

The process of saving the entry location and any associated information that is required in the glossary. In the case of `makeindex` and `xindy`, the entry location, encap, entry item and sort value are written to a supplementary file associated with the glossary that is subsequently read by `makeindex`/`xindy`. In the case of `bib2gls` and the "noidx" method, the entry location, encap and label is written to the `aux` file. With `record=nameref`, the current title and hyperlink target are also included. With `bib2gls`, each line of data in the `aux` is referred to as a "record" and indexing is referred to as "recording". The `record=hybrid` option indexes twice: a `bib2gls` record in the `aux` file and a `makeindex`/`xindy` line in the corresponding file, See §5.8.

Inline full form

The full form of an abbreviation obtained with `\glsxtrfull` or `\glsxtrfullpl` (or case-changing variants). This may or may not produce the same result as the full form on first use of the corresponding `\gls`-like command (the display full form), depending on the abbreviation style.

Inner formatting

The formatting applied by the `innertextformat` option, which redefines `\glsxtrgenentrytextfmt`. The inner formatting can only be applied if `\glsxtrgenentrytextfmt` is embedded within the entry's display style.

Internal field

An internal field may refer to a key that shouldn't be used in the bib file (internal field (bib2gls)), such as the group field, or it may refer to the label used to internally represent the field (which may or may not match the key used to set the field or may not have an associated key), such as useri which corresponds to the user1 key, or it may refer to a field that is only ever used internally that should not be explicitly modified, such as the field used to store the entry's hierarchical level .

Internal field (bib2gls)

A field that is used or assigned by bib2gls that should typically not be used in the bib file.

Internal field label

The field label that forms part of the internal control sequence used to store the field value. This may or may not match the key used to assign the value when defining the entry. See the "Key to Field Mappings" table in the glossaries user manual.

Link text

The text produced by \gls-like and \glstext-like commands that have the potential to be a hyperlink.

Location counter

The counter used to obtain the entry location.

Location encap (format)

A command used to encapsulate an entry location. The control sequence name (without the leading backslash) is identified by the format key. The default encap is \glsnumberformat.

Location list

A list of entry locations (also called a number list). May be suppressed for all glossaries with the package option nonumberlist or for individual glossaries with nonumberlist. With bib2gls, the list may also be suppressed with save-locations=false.

Multi-entry first use

The first time a multi-entry is referenced (or the first time since the multi-entry first use flag was reset). This is not necessary the first use of any of the individual entries that form the set.

Multi-entry first use flag

A conditional that keeps track of whether or not a multi-entry has been used. This is distinct from the first use flags of the individual elements.

Multi-entry subsequent use

When a multi-entry that has already been marked as used is referenced. This is not necessary the subsequent use of any of the individual entries that form the set.

Post-description hook

A hook (\glspostdescription) included in some glossary styles that is used after the description is displayed. The glossaries−extra package modifies this command to provide additional hooks, including category post-description hooks. The glossaries−extra−stylemods package modifies the predefined styles provided with glossaries to ensure that they all use \glspostdescription to allow the category post-description hooks to be implemented.

Post-link hook

A hook (command) that is used after link text to allow code to be automatically added. The base glossaries package provides a general purpose hook \glspostlinkhook. The glossaries−extra package modifies this command to provide additional hooks, including category post-link hooks.

Post-name hook

A hook (command) that is used after the name is displayed in glossary styles. These hooks are implemented by \glossentryname, which needs to be present in the glossary style. The main hook is \glsxtrpostnamehook, which implements auto-indexing (see §12), performs a general purpose hook \glsextrapostnamehook and a category-specific hook \glsxtrpostname⟨*category*⟩.

Print "unsrt" glossary commands (and environment)

The set of commands used for displaying a glossary or partial glossary that have "unsrt" in the name: \printunsrtglossary, \printunsrtglossaries (which internally uses \printunsrtglossary), and \printunsrtinnerglossary. These all simply iterate over the list of entries associated with the given glossary, in the order in which they were added to the glossary (hence "unsrt", which is short for "unsorted"). The way that bib2gls works is that it sorts the entries according to the resource options and adds the entries to the glossary in the required order. These commands may be used with or without bib2gls. If you don't use bib2gls, you will need to manually ensure that the entries are added in the desired order. The printunsrtglossarywrap environment may also be included in this category, although it only sets up the start and end of the glossary for use with \printunsrtinnerglossary.

Resource file

The glstex file created by bib2gls and loaded by \GlsXtrLoadResources (or \glsbibdata).

Resource set

All the settings (resource options) and entries associated with a particular instance of \GlsXtrLoadResources (or \glsbibdata).

Standalone entry

Normally the link text target is created in the glossary, where the entry's name and description (and optionally other information, such as the symbol) are shown. It may be that you don't want a list but need to have the name and description somewhere in the document text. These

are standalone entries. Whilst it is possible to simply use `\glsentryname` and `\glsentrydesc`, the glossaries-extra package provides a way inserting the name that includes the hypertarget for the link text and obeys the post-name hook.

Subsequent use

Using an entry that unsets the first use flag when it has already been unset.

Unicode Transformation Format (8-bit) (UTF-8)

A variable-width encoding that uses 8-bit code units. This means that some characters are represented by more that one byte. XeLaTeX and LuaLaTeX treat the multi-byte sequence as a single token, but the older LaTeX formats have single-byte tokens, which can cause complications, although these have mostly been addressed with the newer kernels introduced over the past few years. Related blog article: Binary Files, Text Files and File Encodings.[4]

Whatsit

A command whose execution is delayed or an OS-specific special command. This includes writing to external files (which is what indexing does).

[4] dickimaw-books.com/blog/binary-files-text-files-and-file-encodings/

Glossary Entry Keys Summary

These are options that can be passed to commands that define entries, such as `\newglossaryentry` or `\newabbreviation`.

access={⟨*text*⟩} (requires accsupp)

Accessibility text corresponding to the `name` field. This field will be automatically set by `\newabbreviation`, if not provided and the `nameshortaccess` attribute is set. See §9.1.

alias={⟨*xr-label*⟩} glossaries−extra §3.2; 34

Behaves in a similar manner to `see={[\seealsoname]⟨xr-label⟩}` but also sets up aliasing which makes the link text hyperlink to ⟨*xr-label*⟩ instead.

category=⟨*category-label*⟩ *initial:* `general` glossaries−extra §3.2; 34

The entry's category (must be a simple label).

counter={⟨*counter-name*⟩}

If set, the value indicates the location counter to use by default when indexing this entry (overrides the counter associated with the glossary or the `counter` package option).

description={⟨*text*⟩}

The entry's description, as displayed in the glossary. If required in the text, use `\glsdesc` (if indexing and hyperlinks are required) or `\glsentrydesc`. Glossary styles should use `\glossentrydesc` and `\glspostdescription` to incorporate the post-description hook.

descriptionaccess={ ⟨*text*⟩ } (requires accsupp)

Accessibility text corresponding to the description field.

descriptionplural={ ⟨*text*⟩ }

The plural form of the entry's description, if applicable. If omitted, this is set to the same value as the description, since descriptions tend not to be a singular entity.

descriptionpluralaccess={ ⟨*text*⟩ } (requires accsupp)

Accessibility text corresponding to the descriptionplural field.

first={ ⟨*first*⟩ }

The entry's text, as displayed on first use of \gls-like commands. Note that using an abbreviation style or post-link hooks is a more flexible approach. If omitted, this value is assumed to be the same as the text key.

firstaccess={ ⟨*text*⟩ } (requires accsupp)

Accessibility text corresponding to the first field. This field will be automatically set by \newabbreviation, if not provided and the firstshortaccess attribute is set. See §9.1.

firstplural={ ⟨*text*⟩ }

The entry's plural form, as displayed on first use of plural \gls-like commands, such as \glspl. If this key is omitted, then the value will either be the same as the plural field, if the first key wasn't used, or the value will be taken from the first key with \glspluralsuffix appended.

firstpluralaccess={ ⟨*text*⟩ } (requires accsupp)

Accessibility text corresponding to the firstplural field. This field will be automatically

set by `\newabbreviation`, if not provided and the `firstshortaccess` attribute is set. See §9.1.

group={⟨*group-label*⟩} glossaries−extra

§3.2; 35

The group label that identifies which group the entry belongs to. This key is only available with the `record=only` and `record=nameref` options, and is set by `bib2gls`, if invoked with `--group` or `-g`. This is an internal key assigned by `bib2gls` as a by-product of sorting. Explicit use without reference to the order of entries can result in fragmented groups. The corresponding title can be set with `\glsxtrsetgrouptitle`, although this is more commonly done implicitly within the `glstex` file.

location={⟨*location-list*⟩} glossaries−extra

§3.2; 36

The formatted location list used by the "unsrt" family of commands. This key is only available with the `record` option and is set by `bib2gls` unless `save-locations`false is set.

long={⟨*long-form*⟩}

A field that is set by `\newabbreviation` to the entry's long (unabbreviated) form. It typically shouldn't be used explicitly with `\newglossaryentry` as `\newabbreviation` makes other modifications to ensure that when the entry is referenced with the `\gls`-like commands, it will obey the appropriate abbreviation style. If you are using `bib2gls` then this field should be used in the `bib` file when defining abbreviations.

longaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `long` field.

longplural={⟨*long-form*⟩}

As `long` but the plural form.

longpluralaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `longplural` field.

name={⟨*text*⟩}

The entry's name, as displayed in the glossary. This typically isn't used outside of the glossary (the `text` and `plural` keys are used instead). However, if there is a need to specifically display the entry name, use `\glsname` (if indexing and hyperlinks are required) or `\glsentryname`. Glossary styles should use `\glossentryname`, which uses `\glsentryname` and incorporates the post-name hooks and related attributes.

parent=⟨*parent-label*⟩

The label of the entry's parent (from which the entry's hierarchical level is obtained).

plural={⟨*text*⟩}

The entry's plural form, as displayed on subsequent use of plural `\gls`-like commands, such as `\glspl`. This should be the appropriate plural form of the value provided by the `text` key. If omitted, this value is assumed to be the value of the `text` key with `\glspluralsuffix` appended.

pluralaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `plural` field. This field will be automatically set by `\newabbreviation`, if not provided and the `textshortaccess` attribute is set. See §9.1.

prefix={⟨*text*⟩} glossaries-prefix v3.14a+

The subsequent use singular prefix.

prefixfirst={⟨*text*⟩} glossaries-prefix v3.14a+

The first use singular prefix.

prefixfirstplural={⟨*text*⟩} glossaries-prefix v3.14a+

The first use plural prefix.

prefixplural={⟨*text*⟩} glossaries−prefix v3.14a+

The subsequent use plural prefix.

see={[⟨*tag*⟩]⟨*xr-list*⟩}

With the base glossaries package this simply triggers an automatic cross-reference with `\glssee`. The glossaries−extra package additionally saves the value. Use `autoseeindex=false` to prevent the automatic cross-reference. The ⟨*tag*⟩ defaults to `\seename` and ⟨*xr-list*⟩ should be a comma-separated list of entries that have already been defined.

seealso={⟨*xr-list*⟩} glossaries−extra

§3.2; 34

Behaves in a similar manner to `see={[\seealsoname]⟨xr-list⟩}`.

short={⟨*short-form*⟩}

A field that is set by `\newabbreviation` (and `\newacronym`) to the entry's short (abbreviated) form. It typically shouldn't be used explicitly with `\newglossaryentry` as `\newabbreviation` makes other modifications to ensure that when the entry is referenced with the `\gls`-like commands, it will obey the appropriate abbreviation style. If you are using `bib2gls` then this field should be used in the `bib` file when defining abbreviations.

shortaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `short` field. This field will be automatically set by `\newabbreviation`, if not provided. See §9.1.

shortplural={⟨*short-form*⟩}

As `short` but the plural form. The default is obtained by appending the abbreviation plural suffix, but this behaviour can be altered by category attributes. See §4 for further details.

shortpluralaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `shortplural` field. This field will be automatically

set by `\newabbreviation`, if not provided. See §9.1.

sort=⟨*value*⟩ *initial:* ⟨*entry name*⟩

Specifies the value to use for sorting (overrides the default). This key is usually required for `xindy` if the `name` key only contains commands (for example, the entry is a symbol), but explicitly using this key in other contexts can break certain sort methods. Don't use the `sort` field with `bib2gls`.[1]

symbol={⟨*symbol*⟩} *initial:* `\relax`

The entry's associated symbol (optional), which can be displayed with `\glssymbol` (if indexing and hyperlinks are required) or with `\glsentrysymbol`.

symbolaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `symbol` field.

symbolplural={⟨*symbol plural*⟩}

The plural form of the `symbol`, if applicable, which can be displayed with `\glssymbolplural` (if indexing and hyperlinks are required) or with `\glsentrysymbolplural`. If omitted, this value is set to the same as the `symbol` key (since symbols usually don't have a plural form).

symbolpluralaccess={⟨*text*⟩} (requires `accsupp`)

Accessibility text corresponding to the `symbolplural` field.

text={⟨*text*⟩}

The entry's text, as displayed on subsequent use of `\gls`-like commands. If omitted, this value is assumed to be the same as the `name` key.

[1] `dickimaw-books.com/gallery/index.php?label=bib2gls-sorting`

textaccess={ ⟨*text*⟩ } (requires accsupp)

Accessibility text corresponding to the text field. This field will be automatically set by \newabbreviation, if not provided and the textshortaccess attribute is set. See §9.1.

type=⟨*glossary-label*⟩ *initial:* \glsdefaulttype

Assigns the entry to the glossary identified by ⟨*glossary-label*⟩.

user1={ ⟨*text*⟩ }

A generic field, which can be displayed with \glsuseri (if indexing and hyperlinks are required) or with \glsentryuseri.

user1access={ ⟨*text*⟩ } glossaries−accsupp v4.45+

Accessibility text corresponding to the user1 field.

user2={ ⟨*text*⟩ }

A generic field, which can be displayed with \glsuserii (if indexing and hyperlinks are required) or with \glsentryuserii.

user2access={ ⟨*text*⟩ } glossaries−accsupp v4.45+

Accessibility text corresponding to the user2 field.

user3={ ⟨*text*⟩ }

A generic field, which can be displayed with \glsuseriii (if indexing and hyperlinks are required) or with \glsentryuseriii.

user3access={ ⟨*text*⟩ } glossaries−accsupp v4.45+

Accessibility text corresponding to the user3 field.

user4={⟨*text*⟩}

A generic field, which can be displayed with `\glsuseriv` (if indexing and hyperlinks are required) or with `\glsentryuseriv`.

user4access={⟨*text*⟩} glossaries−accsupp v4.45+

Accessibility text corresponding to the `user4` field.

user5={⟨*text*⟩}

A generic field, which can be displayed with `\glsuserv` (if indexing and hyperlinks are required) or with `\glsentryuserv`.

user5access={⟨*text*⟩} glossaries−accsupp v4.45+

Accessibility text corresponding to the `user5` field.

user6={⟨*text*⟩}

A generic field, which can be displayed with `\glsuservi` (if indexing and hyperlinks are required) or with `\glsentryuservi`.

user6access={⟨*text*⟩} glossaries−accsupp v4.45+

Accessibility text corresponding to the `user6` field.

Glossary Entry Fields Summary

These are internal fields that don't have a corresponding key.

childcount={⟨*number*⟩}

Used with the save-child-count resource option to store the entry's child count (may also be set with \printnoidxglossary under certain circumstances).

childlist={⟨*entry-label-list*⟩}

Used with the save-child-count resource option to store the entry's children as an etoolbox internal list.

dtlsortgroup

As from glossaries v4.57, used to sort the letter group information with \printnoidxglossary.

indexcounter={⟨*target-name*⟩}

Used with the indexcounter package option and the save-index-counter resource option. The value is set to the hyperlink target of the first wrglossary location or the first instance for a specific location encap.

loclist={⟨*etoolbox list*⟩} glossaries v4.04+

Used by \printnoidxglossary to provide the locations. The value is an etoolbox list of individual locations which are obtained from the aux file. This field will also be used by the "unsrt" family of commands if location isn't set.

recordcount={ ⟨*number*⟩ }

Used with the `--record-count` switch to store the total number of records for the associated entry.

recordcount.⟨*counter*⟩={ ⟨*number*⟩ }

Used with the `--record-count` switch to store the total number of records with the location counter ⟨*counter*⟩ for the associated entry.

recordcount.⟨*counter*⟩**.**⟨*location*⟩={ ⟨*number*⟩ }

Used with the `--record-count-unit` switch to store the total number of records with the location counter ⟨*counter*⟩ set to ⟨*location*⟩ for the associated entry.

record.⟨*counter*⟩={ ⟨*location*⟩ }

Used with `\GlsXtrRecordCounter` to store an etoolbox internal list of locations (without encap) corresponding to the given counter.

secondarygroup={ ⟨*group-label*⟩ }

Used by `bib2gls` to store the group label obtained from the secondary sort.

useri

Corresponds to `user1` key.

userii

Corresponds to `user2` key.

useriii

Corresponds to `user3` key.

\gls-Like and \glstext-Like Options Summary

Most (but not all) of these options can be used in the optional argument of all the \gls-like, \glstext-like and \glsadd commands.

counter=⟨*counter-name*⟩ glossaries

The location counter.

format=⟨*cs-name*⟩ glossaries

The control sequence name (without the leading backslash) that should be used to encapsulate the entry location.

hyper=⟨*boolean*⟩ *default:* true; *initial:* true glossaries

Determines whether or not the link text should have a hyperlink (provided hyperlinks are supported).

hyperoutside=⟨*boolean*⟩ *default:* true; *initial:* true glossaries−extra v1.21+ §5.1.2; 194

Determines whether the hyperlink should be inside or outside of \glstextformat.

innertextformat=⟨*csname*⟩ *initial:* glsxtrdefaultentrytextfmt
glossaries−extra v1.49+ §5.1.2; 195

The name of the control sequence to use for the inner formatting.

local=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries v3.04+

If true use `\glslocalunset` to unset the first use flag, otherwise use `\glsunset` (only applies to `\gls`-like commands).

noindex=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries−extra

§5.1.2; 197

If `true` this option will suppress indexing. If you are using `bib2gls`, you may want to consider using `format=glsignore` to prevent a location but ensure that the entry is selected.

postunset=⟨*value*⟩ *default:* `global`; *initial:* `global` glossaries−extra v1.49+

§5.1.2; 195

Determines whether or not to unset the first use flag after the link text. The value may be one of: `global`, `local` or `none` (only applies to `\gls`-like commands).

prefix=⟨*link-prefix*⟩ glossaries−extra v1.31+

§5.1.2; 200

The prefix to use for the entry's hyperlink target.

prereset=⟨*value*⟩ *default:* `local`; *initial:* `none` glossaries−extra v1.49+

§5.1.2; 195

Determines whether or not to reset the entry before the link text. Allowed values: `none` (no reset), `local` (localise the reset) and `global`.

preunset=⟨*value*⟩ *default:* `local`; *initial:* `none` glossaries−extra v1.49+

§5.1.2; 196

Determines whether or not to unset the entry before the link text. Allowed values: `none` (no unset), `local` (localise the unset) and `global`.

textformat=⟨*csname*⟩ glossaries−extra v1.30+

§5.1.2; 195

The name of the control sequence to use instead of `\glstextformat` to encapsulate the link text.

theHvalue=⟨*the-H-value*⟩ glossaries−extra v1.19+ §5.1.2; 200

Set the hyper location to this value instead of obtaining it from \theH⟨*counter*⟩.

thevalue=⟨*location*⟩ glossaries−extra v1.19+ §5.1.2; 199

Set the location to this value instead of obtaining it from the location counter.

types={⟨*glossary list*⟩} glossaries

Only available with \glsaddall, the value is the list of glossaries to iterate over.

wrgloss=⟨*position*⟩ *initial:* before glossaries−extra v1.14+ §5.1.2; 198

Determines whether to do the indexing before or after the link text. Allowed values: before and after.

Multi-Entry Set Options Summary

all=⟨*option list*⟩ §7.10; 369

Options to pass to the `\gls`-like command for each element.

category=⟨*category-label*⟩ §7.9.6; 368

The category to assign to the multi-entry set.

encapmain=⟨*value*⟩ *initial:* `glsnumberformat` §7.9.2; 365

The value to pass to the `format` option for the main entry.

encapothers=⟨*value*⟩ *initial:* `glsnumberformat` §7.9.2; 365

The value to pass to the `format` option for the "other" elements.

firstprefix=⟨*value*⟩ §7.9.4; 366

The prefix to use on first use of the multi-entry.

firstskipmain=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.9.5; 367

Determines whether or not to skip the main entry on first use.

firstskipothers=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.9.5; 367

Determines whether or not to skip the "other" elements on first use.

firstsuffix=⟨*value*⟩ §7.9.4; 366

The suffix to use on first use of the multi-entry.

hyper=⟨*boolean*⟩ *default:* `true` §7.10; 370

Indicates whether or not to use hyperlinks, if supported, for all elements. This option is for use in the optional argument of `\mgls` and can be set implicitly with the default behaviour of the * and + modifiers.

hyper=⟨*value*⟩ *initial:* `individual` §7.9.6; 367

Indicates which elements should have hyperlinks, if supported. This option is a multi-entry setting, see §7.9.

indexmain=⟨*value*⟩ *default:* `true`; *initial:* `true` §7.9.1; 364

Indicates if the main element should be indexed, should only be indexed on first use or should not indexed.

indexothers=⟨*value*⟩ *default:* `true`; *initial:* `true` §7.9.1; 364

Indicates if the "other" elements should be indexed, should only be indexed on first use or should not indexed.

main=⟨*option list*⟩ §7.10; 369

Options to pass to the `\gls`-like command for the main entry.

mglsopts=⟨*option list*⟩ §7.9.6; 368

The default options to pass to commands like `\mgls`.

mpostlink=⟨*value*⟩ *default:* `true`; *initial:* `true` §7.9.3; 365

Indicates whether or not the multi-entry post-link hook should be enabled and, if so, whether it should only be enabled on first or subsequent use.

mpostlinkelement=⟨*value*⟩ *initial:* `last` §7.9.3; 366

Indicates which post-link hook to use if the multi-entry post-link hook has been enabled.

multiunset=⟨*value*⟩ *initial:* `global` §7.10; 370

Indicates whether or not the multi-entry first use flag should be unset.

others=⟨*option list*⟩ §7.10; 369

Options to pass to the `\gls`-like command for the "other" elements.

postlinks=⟨*value*⟩ *initial:* `none` §7.9.3; 365

Indicates which post-link hooks should be enabled.

presetlocal=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 370

Indicates whether or not the prereset options should have a local or global effect.

resetall=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 370

Indicates whether or not to reset all elements' first use flag before using `\gls`.

resetmain=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 370

Indicates whether or not to reset the main entry's first use flag before using `\gls`.

resetothers=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 370

Indicates whether or not to reset all "other" elements' first use flag before using `\gls`.

setup=⟨*option list*⟩ §7.10; 369

Multi-entry options that will override any conflicting options already assigned to the multi-entry.

textformat=⟨*value*⟩ *initial:* `@firstofone` §7.9.6; 368

The control sequence name of the command that should encapsulate the entire content.

unsetall=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 370

Indicates whether or not to unset all elements' first use flag before using `\gls`.

unsetmain=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 371

Indicates whether or not to unset the main entry's first use flag before using `\gls`.

unsetothers=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.10; 371

Indicates whether or not to unset all "other" elements' first use flag before using `\gls`.

usedprefix=⟨*value*⟩ §7.9.4; 366

The prefix to use on subsequent use of the multi-entry.

usedskipmain=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.9.5; 367

Determines whether or not to skip the main entry on subsequent use.

usedskipothers=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §7.9.5; 367

Determines whether or not to skip the “other” elements on subsequent use.

usedsuffix=⟨*value*⟩ §7.9.4; 366

The suffix to use on subsequent use of the multi-entry.

Print [Unsrt|noidx] Glossary Options Summary

Most (but not all) of these options can be used in the optional argument of all the print glossary commands: `\printglossary`, `\printnoidxglossary`, `\printunsrtglossary` and `\printunsrtinnerglossary`. Some may be used in the optional argument of the printunsrtglossarywrap environment.

entrycounter=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries v4.08+

If true, enable the entry counter.

flatten=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries−extra v1.49+ §8.3; 388

If true, treats all entries as though they have the same hierarchical level (the value of `level-offset`). This option is only available for the "unsrt" family of commands and the printunsrtglossarywrap environment.

groups=⟨*boolean*⟩ *default:* `true`; *initial:* `true` glossaries−extra v1.44+ §8.3; 388

Enables group formation. This option is only available for the "unsrt" family of commands and the printunsrtglossarywrap environment. Note that no groups will be formed when invoking `bib2gls` with the default `--no-group`, regardless of this setting.

label=⟨*label*⟩ glossaries−extra v1.39+ §8.3; 387

Adds `\label{`⟨*label*⟩`}` to the start of the glossary (after the title). Not available with `\printunsrtinnerglossary`.

leveloffset=⟨*offset*⟩ *initial:* 0 glossaries−extra v1.44+ §8.3; 388

Set or increment the hierarchical level offset. If ⟨*offset*⟩ starts with ++ then the current offset is incremented by the given amount otherwise the current offset is set to ⟨*offset*⟩. For example,

an entry with a normal hierarchical level of 1 will be treated as though it has hierarchical level $1 + \langle offset \rangle$. This option is only available for the "unsrt" family of commands and the printunsrtglossarywrap environment.

nogroupskip=⟨*boolean*⟩ *default:* `true`; *initial:* `false`

Suppress the gap implemented by some glossary styles between groups.

nonumberlist=⟨*boolean*⟩ *default:* `true`; *initial:* `false` §8.3; 386

Suppress the location list. Note that `nonumberlist=false` will have no effect with the `save-locations=false` resource option as there won't be any location lists to display.

nopostdot=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries v4.08+

Suppress the post-description punctuation.

numberedsection=⟨*value*⟩ *default:* `nolabel`; *initial:* `false` §8.3; 387

Indicates whether or not glossary section headers will be numbered and also if they should automatically be labelled. The `numberedsection` package option will change the default setting to match.

postamble=⟨*text*⟩ glossaries-extra v1.49+ §8.3; 388

Redefines `\glossarypostamble` to ⟨*text*⟩.

preamble=⟨*text*⟩ glossaries-extra v1.49+ §8.3; 388

Redefines `\glossarypreamble` to ⟨*text*⟩.

prefix=⟨*prefix*⟩ glossaries-extra v1.31+ §8.3; 388

Redefines `\glolinkprefix` to ⟨*prefix*⟩.

sort=⟨*method*⟩ *default:* `standard` §8.3; 386

Only available with `\printnoidxglossary`, this indicates how the glossary should be ordered.

style=⟨*style-name*⟩ §8.3; 387

Use the ⟨*style-name*⟩ glossary style.

subentrycounter=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries v4.08+

If true, enable the sub-entry counter.

target=⟨*boolean*⟩ *default:* `true`; *initial:* `true` glossaries−extra v1.12+ §8.3; 391

If true, each entry in the glossary should have a hypertarget created, if supported by the glossary style and if hyperlinks are enabled.

targetnameprefix=⟨*prefix*⟩ glossaries−extra v1.20+ §8.3; 389

Inserts ⟨*prefix*⟩ at the start of the hypertarget names.

title=⟨*text*⟩ §8.3; 387

Sets the glossary title (overriding the default).

toctitle=⟨*text*⟩ §8.3; 387

Sets the glossary toc title (overriding the default).

type=⟨*glossary-label*⟩ *default:* `\glsdefaulttype`

Identifies the glossary to display.

Abbreviation Styles Summary

Abbreviations defined using \newabbreviation will follow the style associated with the entry's category. If there is no style associated with the entry's category, the style for the abbreviation category is used (the default is long-short). Note that glossaries-extra redefines \newacronym to use \newabbreviation with category=acronym so any entry defined with \newacronym will use the abbreviation style for the acronym category (the default is short-nolong).

footnote-desc *alias:* short-footnote-desc

footnote-em *alias:* short-em-footnote

footnote-sc *alias:* short-sc-footnote

footnote-sm *alias:* short-sm-footnote

footnote *alias:* short-footnote

long-desc-em *alias:* long-noshort-em-desc

long-desc-sc *alias:* long-noshort-sc-desc

long-desc-sm *alias:* long-noshort-sm-desc

long-desc *alias:* long-noshort-desc

long-em-noshort-em-desc-noreg 138

An abbreviation style like long-em-noshort-em-desc but sets the regular attribute to false.

long-em-noshort-em-desc 83

An abbreviation style like long-noshort-desc but formats both the long and short form in an emphasized font (\emph).

long-em-noshort-em-noreg 138

An abbreviation style like long-em-noshort-em but sets the regular attribute to false.

long-em-noshort-em 82

An abbreviation style like long-noshort but formats both the long and short form in an emphasized font (\emph).

long-em-short-em-desc 91

An abbreviation style like long-short-desc but formats both the long and short form in an emphasized font (\emph).

long-em-short-em 91

An abbreviation style like long-short but formats both the long and short form in an emphasized font (\emph).

long-em *alias:* long-noshort-em

long-hyphen-noshort-desc-noreg 113

An abbreviation style like long-noshort-desc but checks if the inserted material starts with a hyphen (use with `markwords` attribute).

long-hyphen-noshort-noreg 114

An abbreviation style like long-hyphen-short-hyphen but doesn't show the short form on first use.

long-hyphen-postshort-hyphen-desc 112

An abbreviation style like long-hyphen-short-hyphen-desc but places the insert and parenthetical material in the post-link hook.

long-hyphen-postshort-hyphen 110

An abbreviation style like long-hyphen-short-hyphen but places the insert and parenthetical material in the post-link hook.

long-hyphen-short-hyphen-desc 111

An abbreviation style like long-hyphen-short-hyphen but the description must be supplied.

long-hyphen-short-hyphen 109

An abbreviation style like long-short but checks if the inserted material starts with a hyphen (use with `markwords` or `markshortwords` attributes).

long-noshort-desc-noreg 138

As long-noshort-desc but it will set the `regular` attribute to `false`.

long-noshort-desc 76

An abbreviation style that only shows the long form on first use and subsequent use. The short form won't be showed unless you use a command like `\glsxtrshort`. The `description` key must be supplied. The full form will only be shown with commands like `\glsxtrfull`. This style sets the `regular` attribute to `true`.

long-noshort-em-desc 82

An abbreviation style like long-noshort-desc but formats the short form in an emphasized font (`\emph`).

long-noshort-em 81

An abbreviation style like long-noshort but formats the short form in an emphasized font (`\emph`).

long-noshort-noreg 138

As long-noshort but it will set the `regular` attribute to `false`.

long-noshort-sc-desc 79

An abbreviation style like long-noshort-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

long-noshort-sc 78

An abbreviation style like long-noshort but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

long-noshort-sm-desc 80

An abbreviation style like long-noshort-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

long-noshort-sm 79

An abbreviation style like long-noshort but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

long-noshort 77

An abbreviation style that only shows the long form on first use and subsequent use. The short form won't be showed unless you use a command like `\glsxtrshort`. The full form will only be shown with commands like `\glsxtrfull`. This style sets the `regular` attribute to `true`.

long-only-short-only-desc 119

An abbreviation style like long-only-short-only but the description must be supplied.

long-only-short-only 118

An abbreviation style that only shows the long form on first use and only shows the short form on subsequent use.

long-only-short-sc-only-desc 121

An abbreviation style like long-only-short-sc-only but the description must be supplied.

long-only-short-sc-only 120

An abbreviation style like long-only-short-only but uses small caps for the short form.

long-postshort-sc-user-desc 97

An abbreviation style like long-postshort-sc-user but the description must be supplied.

long-postshort-sc-user 96

An abbreviation style like long-postshort-user but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

long-postshort-user-desc 95

An abbreviation style like long-postshort-user but the description must be supplied.

long-postshort-user 94

An abbreviation style like long-short-user but the parenthetical content is placed in the post-link hook.

long-sc *alias:* `long-noshort-sc`

long-short-desc 85

As long-short but the `description` must be supplied in ⟨*options*⟩.

long-short-em-desc 90

An abbreviation style like long-short-desc but formats the short form in an emphasized font (`\emph`).

long-short-em 89

An abbreviation style like long-short but formats the short form in an emphasized font (`\emph`).

long-short-sc-desc 87

An abbreviation style like long-short-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

long-short-sc 86

An abbreviation style like long-short but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

long-short-sm-desc 88

An abbreviation style like long-short-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

long-short-sm 88

An abbreviation style like long-short but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

long-short-user-desc 93

An abbreviation style like long-short-user but the description must be supplied.

long-short-user 92

An abbreviation style like long-short but includes the value of the field identified by `\glsxtruserfield` (if set) in the parenthetical content.

long-short 84

An abbreviation style that shows the long form followed by the short form on first use. If the ⟨*insert*⟩ argument is used with the `\gls`-like or `\glstext`-like commands, it will be placed after the long form on first use. On subsequent use, only the short form is shown (followed by ⟨*insert*⟩, if provided). This style sets the `regular` attribute to `false` (which means that the `\gls`-like commands won't use the `first`/`firstplural` or `text`/`plural` values).

long-sm *alias:* `long-noshort-sm`

long *alias:* `long-noshort`

nolong-short-em 75

An abbreviation style like nolong–short but formats the short form in an emphasized font (`\emph`).

nolong-short-noreg 137

As nolong–short but it will set the `regular` attribute to `false`.

nolong-short-sc 71

An abbreviation style like nolong–short but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

nolong-short-sm 73

An abbreviation style like nolong–short but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

nolong-short 69

As short–nolong but the inline full form shows the long form followed by the short form in parentheses.

postfootnote-desc *alias:* `short-postfootnote-desc`

postfootnote-em *alias:* `short-em-postfootnote`

postfootnote-sc *alias:* `short-sc-postfootnote`

postfootnote-sm *alias:* short-sm-postfootnote

postfootnote *alias:* short-postfootnote

short-desc *alias:* short-nolong-desc

short-em-desc *alias:* short-em-nolong-desc

short-em-footnote-desc 134

An abbreviation style like short-footnote-desc but formats the short form in an emphasized font (\emph).

short-em-footnote 133

An abbreviation style like short-footnote but formats the short form in an emphasized font (\emph).

short-em-long-desc 103

An abbreviation style like short-long-desc but formats the short form in an emphasized font (\emph).

short-em-long-em-desc 105

An abbreviation style like short-long-desc but formats both the long and short form in an emphasized font (\emph).

short-em-long-em 104

An abbreviation style like short-long but formats both the long and short form in an emphasized font (\emph).

short-em-long 102

An abbreviation style like short-long but formats the short form in an emphasized font (`\emph`).

short-em-nolong-desc 75

An abbreviation style like short-nolong-desc but formats the short form in an emphasized font (`\emph`).

short-em-nolong 74

An abbreviation style like short-nolong but formats the short form in an emphasized font (`\emph`).

short-em-postfootnote-desc 136

An abbreviation style like short-postfootnote-desc but formats the short form in an emphasized font (`\emph`).

short-em-postfootnote 135

An abbreviation style like short-postfootnote but formats the short form in an emphasized font (`\emph`).

short-em *alias:* `short-em-nolong`

short-footnote-desc 122

As short-footnote but the `description` must be supplied in ⟨*options*⟩.

short-footnote 121

An abbreviation style that shows the short form with the long form as a footnote on first use. If the ⟨*insert*⟩ argument is used with the `\gls`-like or `\glstext`-like commands, it will be placed after the short form, before the footnote marker, on first use. On subsequent use,

only the short form is shown (followed by ⟨*insert*⟩, if provided). The inline full form shows the short form followed by the long form in parentheses. This style sets the `regular` attribute to `false` (which means that the `\gls`-like commands won't use the `first`/`firstplural` or `text`/`plural` values). This style also sets the `nohyperfirst` attribute to `true` to avoid nesting the footnote marker link. If you want hyperlinks on first use, use the short-postfootnote style instead.

short-hyphen-long-hyphen-desc 117

An abbreviation style like short-hyphen-long-hyphen but the description must be supplied.

short-hyphen-long-hyphen 115

An abbreviation style like short-long but checks if the inserted material starts with a hyphen (use with `markwords` or `markshortwords` attributes).

short-hyphen-postlong-hyphen-desc 117

An abbreviation style like short-hyphen-long-hyphen-desc but the insert and parenthetical material are placed in the post-link hook.

short-hyphen-postlong-hyphen 116

An abbreviation style like short-hyphen-long-hyphen but the insert and parenthetical material are placed in the post-link hook.

short-long-desc 99

As short-long but the `description` must be supplied in ⟨*options*⟩.

short-long-user-desc 106

An abbreviation style like short-long-user but the description must be supplied.

short-long-user 105

An abbreviation style like short-long but includes the value of the field identified by `\glsxtr-userfield` (if set) in the parenthetical content.

short-long 98

An abbreviation style that shows the short form followed by the long form on first use. If the ⟨*insert*⟩ argument is used with the `\gls`-like or `\glstext`-like commands, it will be placed after the short form on first use. On subsequent use, only the short form is shown (followed by ⟨*insert*⟩, if provided). This style sets the `regular` attribute to `false` (which means that the `\gls`-like commands won't use the `first`/`firstplural` or `text`/`plural` values).

short-nolong-desc-noreg 137

As short-nolong-desc but it will set the `regular` attribute to `false`.

short-nolong-desc 68

As short-nolong but the `description` must be supplied in ⟨*options*⟩.

short-nolong-noreg 137

As short-nolong but it will set the `regular` attribute to `false`.

short-nolong 67

An abbreviation style that only shows the short form on first use and subsequent use. The long form won't be showed unless you use a command like `\glsxtrlong`. The full form will only be shown with commands like `\glsxtrfull`. This style sets the `regular` attribute to `true`.

short-postfootnote-desc 124

As short-postfootnote but the `description` must be supplied in ⟨*options*⟩.

short-postfootnote 123

Similar to short-footnote but the footnote is placed in the post-link hook.

short-postlong-user-desc 108

An abbreviation style like short-postlong-user but the description must be supplied.

short-postlong-user 107

An abbreviation style like short-long but includes the value of the field identified by `\glsxtr-userfield` (if set) in the parenthetical content, which is placed in the post-link hook.

short-sc-desc *alias:* `short-sc-nolong-desc`

short-sc-footnote-desc 126

An abbreviation style like short-footnote-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-footnote 125

An abbreviation style like short-footnote but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-long-desc 100

An abbreviation style like short-long-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-long 99

An abbreviation style like short-long but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-nolong-desc 70

An abbreviation style like short-nolong-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-nolong 69

An abbreviation style like short-nolong but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-postfootnote-desc 128

An abbreviation style like short-postfootnote-desc but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc-postfootnote 127

An abbreviation style like short-postfootnote but formats the short form in a small caps font (`\textsc`). The short form should therefore be in lowercase.

short-sc *alias:* `short-sc-nolong`

short-sm-desc *alias:* `short-sm-nolong-desc`

short-sm-footnote-desc 130

An abbreviation style like short-footnote-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-footnote 129

An abbreviation style like short-footnote but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-long-desc 102

An abbreviation style like short-long-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-long 101

An abbreviation style like short-long but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-nolong-desc 72

An abbreviation style like short-nolong-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-nolong 72

An abbreviation style like short-nolong but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-postfootnote-desc 132

An abbreviation style like short-postfootnote-desc but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm-postfootnote 131

An abbreviation style like short-postfootnote but formats the short form in a smaller font (`\textsmaller`). The relsize package is must be loaded.

short-sm *alias:* `short-sm-nolong`

short *alias:* `short-nolong`

Glossary Styles Summary

The default style may be set with `\setglossarystyle` or use:

```
\usepackage[stylemods=⟨name⟩,style=⟨style-name⟩]
{glossaries-extra}
```

where the style is provided by package glossary-⟨*name*⟩. The default style can be overridden for individual glossaries with the `style` option. For a summary of all available styles, see Gallery: Predefined Styles.[1]

abbr-long-short glossary-longextra v1.49+ §8.7.2.6; 476

This style displays the glossary in a table for (either longtable or tabular) with two columns: the long form and the short form.

abbr-short-long glossary-longextra v1.49+ §8.7.2.6; 475

This style displays the glossary in a table for (either longtable or tabular) with two columns: the short form and the long form.

altlist glossary-list

As list but starts the description on a new line.

alttree glossary-tree

Like tree but the width of the widest name must be supplied (using a command like `\glssetwidest`).

alttreegroup glossary-tree

As alttree but has headers at the start of each group.

[1] dickimaw-books.com/gallery/index.php?label=glossaries-styles

bookindex glossary−bookindex §8.7.1

This style is designed for indexes. Symbols and descriptions are not shown. Since descriptions aren't shown, there's no post-description hook.

index glossary−tree

A hierarchical style that supports up to level 2, similar to normal indexes, but symbols and descriptions are shown.

indexgroup glossary−tree

As index but has headers at the start of each group.

inline glossary−inline

A compact style with all entries listed in the same paragraph and no groups, locations or symbols.

list glossary−list

This style uses the description environment and places the entry name in the optional argument of `\item`. Symbols and sub-entry names are not shown.

listdotted glossary−list

A list-like style that has a dotted leader between the name and description. The location list isn't shown.

listgroup glossary−list

As list but has headers at the start of each group.

listhypergroup glossary−list

As listgroup but has a row at the start with hyperlinks to each group.

long-booktabs glossary−longbooktabs v4.21+

This style displays the glossary using longtable and horizontal rules from the booktabs package.

long-custom1-name glossary−longextra v1.50+ §8.7.2.7; 480

This style displays the glossary in a table for (either longtable or tabular) with two columns: the custom 1 field and the name.

long-custom2-name glossary−longextra v1.50+ §8.7.2.7; 481

This style displays the glossary in a table for (either longtable or tabular) with three columns: the custom 1 field, custom 2 field, and the name.

long-custom3-name glossary−longextra v1.50+ §8.7.2.7; 481

This style displays the glossary in a table for (either longtable or tabular) with four columns: the custom 1 field, custom 2 field, custom 3 field, and the name.

long-desc-custom1-name glossary−longextra v1.50+ §8.7.2.7; 483

This style displays the glossary in a table for (either longtable or tabular) with three columns: the description, the custom 1 field and the name.

long-desc-custom2-name glossary−longextra v1.50+ §8.7.2.7; 483

This style displays the glossary in a table for (either longtable or tabular) with four columns: the description, the custom 1 field, the custom 2 field and the name.

long-desc-custom3-name glossary−longextra v1.50+ §8.7.2.7; 484

This style displays the glossary in a table for (either longtable or tabular) with five columns: the description, the custom 1 field, the custom 2 field, the custom 3 field and the name.

long-desc-name glossary-longextra v1.37+ §8.7.2.1; 465

This style displays the glossary in a table for (either longtable or tabular) with two columns: the description and the name.

long-desc-sym-name glossary-longextra v1.37+ §8.7.2.2; 467

This style displays the glossary in a table for (either longtable or tabular) with three columns: the description, symbol and name.

long-desc-sym glossary-longextra v1.49+ §8.7.2.5; 473

This style displays the glossary in a table for (either longtable or tabular) with two columns: the description and the symbol.

long-loc-desc-name glossary-longextra v1.37+ §8.7.2.3; 469

This style displays the glossary in a table for (either longtable or tabular) with three columns: the location list, description and name.

long-loc-desc-sym-name glossary-longextra v1.37+ §8.7.2.4; 471

This style displays the glossary in a table for (either longtable or tabular) with four columns: the location list, description, symbol and name.

long-loc-sym-desc-name glossary-longextra v1.37+ §8.7.2.4; 470

This style displays the glossary in a table for (either longtable or tabular) with four columns: the location list, symbol, description and name.

long-name-custom1 glossary-longextra v1.50+ §8.7.2.7; 480

This style displays the glossary in a table for (either longtable or tabular) with two columns: the name and the custom 1 field.

long-name-custom1-desc glossary−longextra v1.50+ §8.7.2.7; 483

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, the custom 1 field and the description.

long-name-custom2 glossary−longextra v1.50+ §8.7.2.7; 480

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, custom 1 field and custom 2 field.

long-name-custom2-desc glossary−longextra v1.50+ §8.7.2.7; 483

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, the custom 1 field, the custom 2 field and the description.

long-name-custom3 glossary−longextra v1.50+ §8.7.2.7; 481

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, custom 1 field, custom 2 field and custom 3 field.

long-name-custom3-desc glossary−longextra v1.50+ §8.7.2.7; 483

This style displays the glossary in a table for (either longtable or tabular) with five columns: the name, the custom 1 field, the custom 2 field, the custom 3 field and the description.

long-name-desc-loc glossary−longextra v1.37+ §8.7.2.3; 468

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, description and location list.

long-name-desc-sym-loc glossary−longextra v1.37+ §8.7.2.4; 469

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, description, symbol and location list.

long-name-desc-sym glossary-longextra v1.37+ §8.7.2.2; 466

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, description and symbol.

long-name-desc glossary-longextra v1.37+ §8.7.2.1; 465

This style displays the glossary in a table for (either longtable or tabular) with two columns: the name and the description.

long-name-sym-desc-loc glossary-longextra v1.37+ §8.7.2.4; 470

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, symbol, description and location list.

long-name-sym-desc glossary-longextra v1.37+ §8.7.2.2; 467

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, symbol and description.

long-sym-desc-name glossary-longextra v1.37+ §8.7.2.2; 467

This style displays the glossary in a table for (either longtable or tabular) with three columns: the symbol, description and name.

long-sym-desc glossary-longextra v1.49+ §8.7.2.5; 472

This style displays the glossary in a table for (either longtable or tabular) with two columns: the symbol and the description.

long glossary-long

This style uses the longtable environment (provided by the longtable package). Symbols and sub-entry names are not shown.

longheader glossary−long

This style uses the longtable environment (provided by the longtable package) with a header row. Symbols and sub-entry names are not shown.

longragged glossary−longragged

This style displays the glossary using longtable with ragged right paragraph formatting for the description column.

mcolindex glossary−mcols

As index but puts the content inside a multicols environment.

mcolindexgroup glossary−mcols

As mcolindex but has headers at the start of each group.

mcoltree glossary−mcols

As tree but puts the content inside a multicols environment.

super glossary−super

This style displays the glossary using supertabular.

superragged glossary−superragged

This style displays the glossary using supertabular with ragged right paragraph formatting for the description column.

table glossary−table

§8.7.4; 491

This style is specific to `\printunsrttable`.

topic glossary−topic

§8.7.3; 484

This style is designed for hierarchical glossaries where the top-level entry represents a topic.

topicmcols glossary−topic

§8.7.3; 484

Similar to topic but the sub-entries are placed in a multicols environment.

tree glossary−tree

A hierarchical style that supports unlimited levels (although a deep hierarchy may not fit the available line width) with that shows symbols and descriptions.

treegroup glossary−tree

As tree but has headers at the start of each group.

treehypergroup glossary−tree

As treegroup but has a row at the start with hyperlinks to each group.

treenoname glossary−tree

Like tree but the child entries don't have their name shown.

Command Summary

Symbols

\@glsxtr@doglossary glossaries−extra v1.08+ §8.4.3; 412

Internal command used within the construction of the glossary code by the "unsrt" family of commands. Should not be used or modified but `\printunsrtglossarypredoglossary` can be defined to show the definition of this command for debugging purposes.

\@glsxtr@multientry { ⟨*options*⟩ } { ⟨*multi-label*⟩ } { ⟨*main-label*⟩ } { ⟨*list*⟩ }
glossaries−extra v1.48+ §7.13; 380

Information in the `aux` about a multi-label defined in the previous LaTeX run.

\@glsxtr@org@@starttoc { ⟨*toc*⟩ } §5.3.3; 230

Set to the definition of `\@starttoc` when glossaries−extra loads.

\@glsxtr@org@markboth { ⟨*left text*⟩ } { ⟨*right text*⟩ } §5.3.3; 230

Set to the definition of `\markboth` when glossaries−extra loads.

\@glsxtr@org@markright { ⟨*text*⟩ } §5.3.3; 230

Set to the definition of `\markright` when glossaries−extra loads.

\@glsxtrinmark glossaries−extra v1.07+ §5.3.3; 230

Redefines `\glsxtrifinmark` to do its first argument (⟨*true*⟩).

\@glsxtrnotinmark glossaries−extra v1.07+ §5.3.3; 231

Redefines \glsxtrifinmark to do its first argument (⟨*true*⟩).

A

\AB [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGLS defined by the shortcuts=abbreviations package option.

\Ab [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGls defined by the shortcuts=abbreviations package option.

\ab [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cgls defined by the shortcuts=abbreviations package option.

\abbreviationsname *initial:* Abbreviations glossaries−extra §2.1; 10
(language-sensitive)

Expands to the title of the abbreviations glossary. The default is "Abbreviations" or \acronymname if babel has been detected.

\abbrvpluralsuffix *initial:* \glsxtrabbrvpluralsuffix 176

Style-sensitive abbreviation suffix. This is the command that's actually used in the value of the shortplural key when an entry is defined with \newabbreviation (unless suppressed with the noshortplural attribute). This command is redefined by the abbreviation styles to \glsxtrabbrvpluralsuffix or the style's custom suffix command (such as \glsxtrscsuffix).

\ABP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGLSpl defined by the shortcuts=abbreviations package op-

tion.

\Abp [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGlspl defined by the shortcuts=abbreviations package option.

\abp [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cglspl defined by the shortcuts=abbreviations package option.

\AC [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGLS defined by the shortcuts=ac package option.

\Ac [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGls or \Gls defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ac [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cgls or \gls defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACF [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrfull defined by the shortcuts=ac package option.

\Acf [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrfull or \Acrfull defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acf [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrfull or \acrfull defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACFP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrfullpl defined by the shortcuts=ac package option.

\Acfp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrfullpl or \Acrfullpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acfp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrfullpl or \acrfullpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACL [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrlong defined by the shortcuts=ac package option.

\Acl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrlong or \Acrlong defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrlong or \acrlong defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACLP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrlongpl defined by the shortcuts=ac package option.

\Aclp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrlongpl or \Acrlongpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\aclp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrlongpl or \acrlongpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGLSpl defined by the shortcuts=ac package option.

\Acp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cGlspl or \glspl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \cglspl or \glspl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

⊘ **\Acrfull** [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \acrfull but sentence case.

⊘ **\acrfull** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the full form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with \newabbreviation.

⊘ **\Acrfullpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \acrfullpl but sentence case.

⊘ **\acrfullpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the plural full form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with \newabbreviation.

⊘ **\Acrlong** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \acrlong but sentence case.

⊘ **\acrlong** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the long form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with \newabbreviation.

⊘ **\Acrlongpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As \acrlong but sentence case.

⊘ **\acrlongpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the plural long form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with \newabbreviation.

\acronymfont { ⟨*text*⟩ } glossaries v1.0+

Used to encapsulate the acronym short form on subsequent use by the base glossaries package. This is redefined by glossaries−extra to use `\glsabbrvfont`.

\acronymname *initial:* `Acronyms` glossaries
(language-sensitive)

Expands to the title of the `acronym` glossary.

\acronymtype *initial:* `\glsdefaulttype` glossaries

§4.1.4; 45

Expands to the label of the default acronym glossary. The `acronym` or `acronyms` package option will redefine this to `acronym`. The `abbreviations` package option will redefine this to `\glsxtrabbrvtype` if `acronyms`/`acronym` isn't used.

⊘ **\acrpluralsuffix**

Used for the plural suffixes for the base package's acronym mechanism. Not used with glossaries −extra's abbreviations, which use `\glsxtrabbrvpluralsuffix`, `\abbrvpluralsuffix` and commands provided for use with particular abbreviation styles. This command should not be used with glossaries−extra.

⊘ **\Acrshort** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As `\acrshort` but sentence case.

⊘ **\acrshort** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the short form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with `\newabbreviation`.

⊘ **\Acrshortpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

As `\acrshort` but sentence case.

⊘ **\acrshortpl** [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

Displays the plural short form of an acronym. Only for use with the base glossaries package's acronym mechanism. This command is not compatible with \newabbreviation.

\ACS [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrshort defined by the shortcuts=ac package option.

\Acs [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrshort or \Acrshort defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acs [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrshort or \acrshort defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACSP [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrshortpl defined by the shortcuts=ac package option.

\Acsp [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrshortpl or \Acrshortpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\acsp [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrshortpl or \acrshortpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\AF [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrfull defined by the shortcuts=abbreviations package option.

\Af [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrfull defined by the shortcuts=abbreviations package option.

\af [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrfull defined by the shortcuts=abbreviations package option.

\AFP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrfullpl defined by the shortcuts=abbreviations package option.

\Afp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrfullpl defined by the shortcuts=abbreviations package option.

\afp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrfullpl defined by the shortcuts=abbreviations package option.

\AL [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrlong defined by the shortcuts=abbreviations package option.

\Al [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrlong defined by the shortcuts=abbreviations package option.

\al [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrlong defined by the shortcuts=abbreviations package option.

\ALP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrlongpl defined by the shortcuts=abbreviations package option.

\Alp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrlongpl defined by the shortcuts=abbreviations package option.

\alp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrlongpl defined by the shortcuts=abbreviations package option.

\Alpha glossaries-extra-bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{A}.

\alsoname *initial:* see also (language-sensitive)

Used as a cross-reference tag (provided by language packages, such as babel).

\andname *initial:* \&

Used by \glsseelastsep (provided by glossaries if not already defined).

\apptoglossarypreamble [⟨*type*⟩] {⟨*text*⟩} glossaries−extra v1.12+ §8.2; 386

Appends (locally) ⟨*text*⟩ to the preamble for the glossary identified by ⟨*type*⟩. If ⟨*type*⟩ is omitted, \glsdefaulttype is assumed.

\AS [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrshort defined by the shortcuts=abbreviations package option.

\As [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrshort defined by the shortcuts=abbreviations package option.

\as [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \glsxtrshort defined by the shortcuts=abbreviations package option.

\ASP [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \GLSxtrshortpl defined by the shortcuts=abbreviations package option.

\Asp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3.2; Table 4.1

A synonym for \Glsxtrshortpl defined by the shortcuts=abbreviations package option.

\asp [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ — §4.3.2; Table 4.1

A synonym for \glsxtrshortpl defined by the shortcuts=abbreviations package option.

B

\Beta glossaries−extra−bib2gls v1.27+ — §11.6.8; 622

Defined with \providecommand, this just does \mathrm{B}.

\bibgls@input {⟨*filename*⟩} glossaries−extra v1.49+

Indicates that the bib2gls records are in the file identified in the argument ⟨*filename*⟩, which corresponds to the file ⟨*basename*⟩.aux identified in the option bibglsaux=⟨*basename*⟩.

\bibglscopytoglossary {⟨*entry-label*⟩} {⟨*glossary-type*⟩} bib2gls v3.3+

Command definition provided in the glstex file used with the copy-to-glossary option to copy an entry to another glossary.

\bibglsdualprefixlabel {⟨*label-prefix*⟩} bib2gls v1.8+ — §11.6.7; 620

Hook written to the glstex file identifying the dual label prefix.

\bibglsnewdualindexsymbolsecondary {⟨*label*⟩} {⟨*options*⟩} {⟨*name*⟩} {⟨*description*⟩} bib2gls

Defines secondary terms provided with @dualindexsymbol.

\BibGlsNoCaseChange {⟨*text*⟩} bib2gls v4.1+

Prevents case-changing within bib2gls but simply expands to ⟨*text*⟩ within the LaTeX document.

\BibGlsOptions{⟨*options*⟩} glossaries−extra v1.54+ §11; 540

Supply global bib2gls options (instead of using command line switches).

\bibglsprimaryprefixlabel{⟨*label-prefix*⟩} bib2gls v1.8+ §11.6.7; 620

Hook written to the glstex file identifying the primary label prefix.

\bibglssetlastgrouptitle{⟨*cs*⟩}{⟨*specs*⟩} bib2gls

Sets the last group title.

\bibglssetlocationrecordcount{⟨*entry-label*⟩}{⟨*counter*⟩}
{⟨*location*⟩}{⟨*value*⟩} bib2gls v1.1+ §11.5.2; 578

Sets the location record count for the given entry.

\bibglssetrecordcount{⟨*entry-label*⟩}{⟨*counter*⟩}{⟨*value*⟩}
bib2gls v1.1+ §11.5.2; 578

Sets the ⟨*counter*⟩ record count for the given entry.

\bibglssettotalrecordcount{⟨*entry-label*⟩}{⟨*value*⟩} bib2gls v1.1+ §11.5.2; 578

Sets the total record count for the given entry.

\bibglstertiaryprefixlabel{⟨*label-prefix*⟩} bib2gls v1.8+ §11.6.7; 620

Hook written to the glstex file identifying the tertiary label prefix.

C

\capitalisefmtwords { ⟨*text*⟩ } mfirstuc v2.03+

Converts ⟨*text*⟩ to title case, where ⟨*text*⟩ may contain text-block commands. The starred form only permits a text-block command at the start of the argument. Limitations apply, see the mfirstuc documentation for further details, either:

```
texdoc mfirstuc
```

or visit `ctan.org/pkg/mfirstuc`.

\capitalisewords { ⟨*text*⟩ } mfirstuc v1.06+

Converts ⟨*text*⟩ to title case. Limitations apply, see the mfirstuc documentation for further details, either:

```
texdoc mfirstuc
```

or visit `ctan.org/pkg/mfirstuc`.

\CAT §11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\CAT`.

\cGLS [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §6.1; 321

Like `\GLS` but hooks into the entry counting mechanism.

\cGls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §6.1; 320
glossaries v4.14+

Like `\Gls` but hooks into the entry counting mechanism.

\cgls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §6.1; 319
glossaries v4.14+

Like `\gls` but hooks into the entry counting mechanism.

\cGLSformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }

§6.1; 321

Format used by \cGLS if the entry was not used more than the given trigger value on the previous run.

\cGlsformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries v4.14+

§6.1; 320

Format used by \cGls if the entry was not used more than the given trigger value on the previous run.

\cglsformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries v4.14+

§6.1; 319

Format used by \cgls if the entry was not used more than the given trigger value on the previous run.

\cGLSpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

§6.1; 321

Like \GLSpl but hooks into the entry counting mechanism.

\cGlspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v4.14+

§6.1; 320

Like \Glspl but hooks into the entry counting mechanism.

\cglspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v4.14+

§6.1; 320

Like \glspl but hooks into the entry counting mechanism.

\cGLSplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }

§6.1; 321

Format used by \cGLSpl if the entry was not used more than the given trigger value on the previous run.

\cGlsplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries v4.14+ §6.1; 320

Format used by \cGlspl if the entry was not used more than the given trigger value on the previous run.

\cglsplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries v4.14+ §6.1; 320

Format used by \cglspl if the entry was not used more than the given trigger value on the previous run.

\Chi glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{X}.

\CS §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \CS.

\cs { ⟨*csname*⟩ } §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \csname.

\csGlsXtrLetField { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*cs-name*⟩ }
glossaries−extra v1.12+ §3.5; 41

Like \GlsXtrLetField but internally uses (etoolbox's) \csletcs instead of \cslet.

\currentglossary glossaries v3.0+

Defined by the \print...glossary commands to the current glossary label.

\CustomAbbreviationFields §4.5.3.1; 170

Expands to the default field definitions for the entry.

D

\defglsentryfmt [⟨*type*⟩] {⟨*display*⟩} glossaries

Overrides the default display format (\glsentryfmt) for the given glossary. If ⟨*type*⟩ is omitted, \glsdefaulttype is assumed. This will make the \gls-like commands do ⟨*display*⟩ for any entries that have the type field set to the given ⟨*type*⟩. If you want to support any abbreviation styles, you need to include \glssetabbrvfmt in ⟨*display*⟩. Non-regular abbreviation styles are designed to work with \glsxtrgenabbrvfmt.

\delimN glossaries

Used as a separator between locations.

\delimR glossaries

Used between the start and end of a location range.

\descriptionname *initial:* Description glossaries
(language-sensitive)

Expands to the title of the description column for headed tabular-like styles.

\dGLS [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §11.6.7; 614
glossaries-extra-bib2gls v1.37+

As \dgls but uses \GLS.

\dGls [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §11.6.7; 614
glossaries-extra-bib2gls v1.37+

As \dgls but uses \Gls.

\dgls [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §11.6.7; 613
glossaries-extra-bib2gls v1.37+

Does \gls[⟨*options*⟩]{⟨*prefix*⟩}{entry-label}}[⟨*insert*⟩] for the first prefix in the

prefix list that matches a defined entry.

\dGlsdisp [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*text*⟩} *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra-bib2gls v1.49+ §11.6.7; 614

As \dglsdisp but applies sentence case.

\dglsdisp [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*text*⟩} *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra-bib2gls v1.37+ §11.6.7; 614

As \dgls but uses \glsdisp.

\dGLSfield [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*field-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra-bib2gls v1.49+ §11.6.7; 615

As \dglsfield but all caps.

\dGlsfield [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*field-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra-bib2gls v1.49+ §11.6.7; 615

As \dglsfield but applies sentence case.

\dglsfield [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*field-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra-bib2gls v1.49+ §11.6.7; 615

As \dgls but selects the first matching label that has an entry with the field set.

\dglsfieldactualfieldlabel glossaries-extra-bib2gls v1.49+ §11.6.7; 617

Set by the \dglsfield family of commands to the actual field used. This will either be the requested field or the fallback field.

\dglsfieldcurrentfieldlabel glossaries-extra-bib2gls v1.49+ §11.6.7; 616

Set by the \dglsfield family of commands to the given ⟨*field-label*⟩.

\dglsfieldfallbackfieldlabel *initial:* `text`
glossaries–extra–bib2gls v1.49+

§11.6.7; 617

Expands to the fallback field to use for the `\dglsfield` family of commands.

\dGlslink [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*text*⟩ } *modifiers:* * + ⟨*alt-mod*⟩
glossaries–extra–bib2gls v1.49+

§11.6.7; 614

As `\dglslink` but applies sentence case.

\dglslink [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*text*⟩ } *modifiers:* * + ⟨*alt-mod*⟩
glossaries–extra–bib2gls v1.37+

§11.6.7; 614

As `\dgls` but uses `\glslink`.

\dGLSpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries–extra–bib2gls v1.37+

§11.6.7; 614

As `\dgls` but uses `\GLSpl`.

\dGlspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries–extra–bib2gls v1.37+

§11.6.7; 614

As `\dgls` but uses `\Glspl`.

\dglspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries–extra–bib2gls v1.37+

§11.6.7; 614

As `\dgls` but uses `\glspl`.

\Digamma glossaries–extra–bib2gls v1.27+

§11.6.8; 622

Defined with `\providecommand`, this just does `\mathrm{F}`.

\doifglossarynoexistsordo { ⟨*glossary-type*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries v4.19+

If the glossary given by ⟨*glossary-type*⟩ doesn't exist, this does ⟨*true*⟩, otherwise it generates an error and does ⟨*false*⟩. This uses the starred form of `\ifglossaryexists` to test for existence.

\DTLformatlist { ⟨*csv-list*⟩ } datatool−base v2.28+

Formats the comma-separated list ⟨*csv-list*⟩. One-level expansion is performed on ⟨*csv-list*⟩. See the datatool documentation for further details, either:

```
texdoc datatool
```

or visit `ctan.org/pkg/datatool`.

\DTLifinlist { ⟨*element*⟩ } { ⟨*csv-list*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } datatool−base

Does ⟨*true*⟩ if ⟨*element*⟩ is contained in the comma-separated list ⟨*csv-list*⟩, otherwise does ⟨*false*⟩. One-level expansion is performed on ⟨*csv-list*⟩, but not on ⟨*element*⟩. See the datatool documentation for further details, either:

```
texdoc datatool
```

or visit `ctan.org/pkg/datatool`.

\DTLsortwordlist { ⟨*clist-var*⟩ } { ⟨*handler-cs*⟩ } datatool−base v3.0+

Sorts the given comma-separated list variable, where the values are preprocessed by the given sort handler function. See the datatool documentation for further details, either:

```
texdoc datatool
```

or visit `ctan.org/pkg/datatool`.

E

\eglssetwidest [⟨*level*⟩] { ⟨*name*⟩ } glossaries−extra−stylemods v1.05+ §8.6.5.4; 450

As \glssetwidest but expands ⟨*text*⟩.

\eglsupdatewidest [⟨*level*⟩] { ⟨*name*⟩ } glossaries−extra−stylemods v1.23+ §8.6.5.4; 450

As \glsupdatewidest but expands ⟨*name*⟩.

\eGlsXtrSetField{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+ §3.5; 41

As \GlsXtrSetField but expands the value.

\entryname *initial:* Notation glossaries
(language-sensitive)

Expands to the title of the name column for headed tabular-like styles.

\Epsilon glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{E}.

\Eta glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{H}.

\ExtraCustomAbbreviationFields glossaries−extra v1.31+ §4.5.3.1; 170

Expands to additional fields that need to be set with \newabbreviation.

F

\firstacronymfont { ⟨*text*⟩ } glossaries v1.14+

Used to encapsulate the acronym short form on first use by the base glossaries package. This is redefined by glossaries−extra to use \glsfirstabbrvfont.

\FIRSTLC §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \FIRSTLC.

\FIRSTUC §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \FIRSTUC.

\forallabbreviationlists { ⟨*cs*⟩ } { ⟨*body*⟩ } glossaries−extra v1.42+ §8; 385

Iterates overall all lists of abbreviations, defines the command ⟨*cs*⟩ to the current label and does ⟨*body*⟩.

⊘ **\forallacronyms** { ⟨*cs*⟩ } { ⟨*body*⟩ } glossaries v4.08+

Iterates overall all glossaries that have been declared lists of acronyms, defines the command ⟨*cs*⟩ to the current label and does ⟨*body*⟩. Use \forallabbreviationlists with glossaries−extra.

\forallglossaries [⟨*types*⟩] { ⟨*cs*⟩ } { ⟨*body*⟩ } glossaries

Iterates overall all the glossary labels given in the ⟨*types*⟩ argument, defines the command ⟨*cs*⟩ to the current label and does ⟨*body*⟩. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\forallglsentries [⟨*types*⟩] { ⟨*cs*⟩ } { ⟨*body*⟩ } glossaries

Does \forglsentries for each glossary. The optional argument ⟨*types*⟩ is a comma-separated list of glossary labels. If omitted, all non-ignored glossaries is assumed.

\forglsentries [⟨*type*⟩] {⟨*cs*⟩} {⟨*body*⟩} glossaries

Iterates over all entries in the given glossary and, at each iteration, defines the command ⟨*cs*⟩ to the current entry label and does ⟨*body*⟩. The optional argument ⟨*type*⟩ is the glossary label and defaults to `\glsdefaulttype` if omitted. This command can't be used with `bib2gls` since there are no defined entries until `bib2gls` has selected them and added them to the `glstex` file.

G

\gglssetwidest [⟨*level*⟩] {⟨*name*⟩} glossaries−extra−stylemods v1.21+ §8.6.5.4; 450

As `\glssetwidest` but global.

\gglsupdatewidest [⟨*level*⟩] {⟨*name*⟩} glossaries−extra−stylemods v1.23+ §8.6.5.4; 450

As `\glsupdatewidest` but global.

\gGlsXtrSetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}
glossaries−extra v1.12+ §3.5; 41

As `\GlsXtrSetField` but globally assigns the value.

Glo

\glolinkprefix *initial:* `glo:` glossaries

Expands to the default prefix for the entry's hypertarget anchor in the glossary.

\GlossariesAbbrStyleTooComplexWarning glossaries−extra v1.49+

Issues a warning with `\GlossariesExtraWarning` when a command is used that isn't supported by a complex abbreviation style.

\GlossariesExtraInfo{⟨*message*⟩} glossaries−extra v1.51+

Writes an information message to the transcript.

\glossariesextrasetup{⟨*options*⟩} §2; 9

Change allowed options that are defined or modified by the glossaries−extra package. Note that some options can only be passed as package options.

\GlossariesExtraWarning{⟨*message*⟩}

Writes a warning in the transcript with the current line number. The `nowarn` option redefines this command to do nothing.

\GlossariesExtraWarningNoLine{⟨*message*⟩} glossaries−extra

Writes a warning in the transcript without a corresponding line number. The `nowarn` option redefines this command to do nothing.

\glossaries_if_field_eq:nnN*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*value-tl-var*⟩ {⟨*true*⟩} {⟨*false*⟩}
\glossaries_if_field_eq_p:nnN {⟨*entry-label*⟩} {⟨*field-label*⟩} ⟨*value-tl-var*⟩ §5.16; 315
glossaries−extra v1.55+

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the same value as the given token list variable.

\glossaries_if_field_eq:nnn*TF* {⟨*entry-label*⟩} {⟨*field-label*⟩} {⟨*value-tl*⟩} {⟨*true*⟩} {⟨*false*⟩} §5.16; 315
glossaries−extra v1.55+

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the given value.

\glossaries_if_field_eq_field:nnn*TF* { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } ⟨*field2-label*⟩ { ⟨*true*⟩ } { ⟨*false*⟩ }
\glossaries_if_field_eq_field_p:nnn { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } ⟨*field2-label*⟩
glossaries−extra v1.55+

§5.16; 315

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ that has the same value as the second field identified by ⟨*field2-label*⟩ (for the same entry).

\glossaries_if_field_eq_field:nnnn*TF* { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*entry2-label*⟩ } ⟨*field2-label*⟩ { ⟨*true*⟩ } { ⟨*false*⟩ }
\glossaries_if_field_eq_field_p:nnnn { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*entry2-label*⟩ } ⟨*field2-label*⟩
glossaries−extra v1.55+

§5.16; 316

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ and the entry identified by ⟨*entry2-label*⟩ exists and has an internal field identified by ⟨*field2-label*⟩ and both field values are the same.

\glossaries_if_field_exists:nn*TF* { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
\glossaries_if_field_exists_p:nn { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.55+

§5.16; 315

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ (which may or may not be empty).

\glossaries_if_field_set:nn*TF* { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
\glossaries_if_field_set_p:nn { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.55+

§5.16; 315

True if the entry identified by ⟨*entry-label*⟩ exists and has an internal field identified by ⟨*field-label*⟩ set to a value that is not empty and not \relax.

\glossaries_use_field:nn { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.55+

§5.16; 316

Expands to the value of the field identified by its internal field label for the entry identifed by ⟨*entry-label*⟩ (produces an error if either the field or entry are undefined).

\glossaryentrynumbers { ⟨*location list*⟩ } glossaries

Used within the glossary to encapsulate the location list (redefined by the `nonumberlist` option).

\glossaryheader glossaries

Inserted after `\begin{theglossary}`. This command should be redefined by the glossary style.

\glossaryname *initial:* `Glossary` (language-sensitive)

Expands to the default glossary title (provided by glossaries if not already defined).

\glossarypostamble glossaries

Used at the end of the glossary.

\glossarypreamble glossaries

Used at the start of the glossary. This will be locally redefined to the preamble associated with the current glossary, if one has been set.

\glossarysection [⟨*toc-title*⟩] { ⟨*title*⟩ } glossaries

Occurs at the start of a glossary (except with `\printunsrtinnerglossary`). This will typically be defined to use a sectioning command, such as `\section` or `\chapter`. The default definition follows the `section` and `numberedsection` options.

\glossarytitle glossaries

Initialised by the `\print...glossary` set of commands (such as `\printglossary`) to the current glossary's title.

\glossarytoctitle glossaries

Initialised by the \print...glossary set of commands (such as \printglossary) to the current glossary's table of contents title.

\glossentry{ ⟨*entry label*⟩ } { ⟨*location list*⟩ } glossaries v3.08+

Used to format a top-level entry. This command should be redefined by the glossary style.

\Glossentrydesc{ ⟨*entry-label*⟩ } glossaries v3.08a+

As \glossentrydesc but sentence case.

\glossentrydesc{ ⟨*entry-label*⟩ } glossaries v3.08a+

Used by glossary styles to display the entry's description.

\GLOSSentryname{ ⟨*entry-label*⟩ } glossaries v4.59+

As \glossentryname but all uppercase.

\GlossEntryName{ ⟨*entry-label*⟩ } glossaries v4.59+

As \glossentryname but title case.

\Glossentryname{ ⟨*entry-label*⟩ } glossaries v3.08a+

As \glossentryname but sentence case.

\glossentryname{ ⟨*entry-label*⟩ } glossaries v3.08a+

Used by glossary styles to display the entry's name.

\GLOSSentrynameother{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.6+ §8.6; 436

Behaves like \GLOSSentryname but uses the given field (identified by its internal label) instead of name.

\GlossEntryNameOther{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.6+ §8.6; 436

Behaves like \GlossEntryName but uses the given field (identified by its internal label) instead of name.

\Glossentrynameother{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.54+ §8.6; 436

Behaves like \Glossentryname but uses the given field (identified by its internal label) instead of name.

\glossentrynameother{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.22+ §8.6; 435

Behaves like \glossentryname but uses the given field (identified by its internal label) instead of name.

\Glossentrysymbol{⟨*entry-label*⟩} glossaries v3.08a+

As \glossentrysymbol but sentence case.

\glossentrysymbol{⟨*entry-label*⟩} glossaries v3.08a+

Used by glossary styles to display the entry's symbol.

\glossxtrsetpopts glossaries−extra v1.07+ §5.4; 235

Used at the start of each glossary to set the current options for the \glsxtrp set of commands (with \glsxtrsetpopts).

Gls

\GLS [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries

As \gls but converts the link text to all caps.

\Gls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries

As \gls but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\gls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced depends on whether or not this is the first use and whether or not the regular attribute has been set. The ⟨*insert*⟩ argument may be inserted at the end of the link text or may be inserted at a different point (for example, after the long form on first use for some abbreviation styles. For the first optional argument, see \glslink options.

\glsabbrvdefaultfont { ⟨*text*⟩ } 139

Formatting command for the short form used by the abbreviation styles that don't apply a font change by default.

\glsabbrvemfont { ⟨*text*⟩ } glossaries−extra v1.04+ 163

Short form font used by the "em" abbreviation styles.

\glsabbrvfont { ⟨*text*⟩ } 177

Font formatting command for the short form, initialised by the abbreviation style.

\glsabbrvhyphenfont { ⟨*text*⟩ } glossaries−extra v1.17+ 154

Short form font used by the "hyphen" abbreviation styles.

\glsabbrvonlyfont{⟨*text*⟩} glossaries−extra v1.17+ 159

Short form font used by the "only" abbreviation styles.

\glsabbrvscfont{⟨*text*⟩} glossaries−extra v1.17+ 162

Short form font used by the small caps "sc" abbreviation styles.

\glsabbrvsconlyfont{⟨*text*⟩} glossaries−extra v1.48+ 160

Short form font used by the "sc-only" styles, such as long−only−short−sc−only.

\glsabbrvscuserfont{⟨*text*⟩} glossaries−extra v1.48+ 143

Short form font used by the small caps "sc-user" abbreviation styles.

\glsabbrvsmfont{⟨*text*⟩} glossaries−extra v1.17+ 163

Short form font used by the "sm" abbreviation styles.

\glsabbrvuserfont{⟨*text*⟩} glossaries−extra v1.04+ 142

Short form font used by the "user" abbreviation styles.

\glsabspace{⟨*label*⟩} glossaries−extra v1.49+

As `\glsacspace` but includes inner formatting. Unlike `\glsacspace`, this command is robust.

\GLSaccessdesc{⟨*entry-label*⟩} §9.2; 513

The all caps version of `\glsaccessdesc`.

\Glsaccessdesc{⟨*entry-label*⟩} §9.2; 513

The sentence case version of \glsaccessdesc.

\glsaccessdesc{⟨*entry-label*⟩} §9.2; 512

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the description key with the accessibility support enabled for that key (descriptionaccess). If there is no accessibility support, this just uses \glsentrydesc.

\GLSaccessdescplural{⟨*entry-label*⟩} §9.2; 513

The all caps version of \glsaccessdescplural.

\Glsaccessdescplural{⟨*entry-label*⟩} §9.2; 513

The sentence case version of \glsaccessdescplural.

\glsaccessdescplural{⟨*entry-label*⟩} §9.2; 513

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the descriptionplural key with the accessibility support enabled for that key (descriptionpluralaccess). If there is no accessibility support, this just uses \glsentrydescplural.

\GLSaccessfirst{⟨*entry-label*⟩} §9.2; 511

The all caps version of \glsaccessfirst.

\Glsaccessfirst{⟨*entry-label*⟩} §9.2; 511

The sentence case version of \glsaccessfirst.

\glsaccessfirst{⟨*entry-label*⟩} §9.2; 511

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the first key with the accessibility support enabled for that key (firstaccess). If there is no accessibility support, this just uses \glsentryfirst.

\GLSaccessfirstplural{⟨*entry-label*⟩} §9.2; 512

The all caps version of \glsaccessfirstplural.

\Glsaccessfirstplural{⟨*entry-label*⟩} §9.2; 512

The sentence case version of \glsaccessfirstplural.

\glsaccessfirstplural{⟨*entry-label*⟩} §9.2; 511

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the firstplural key with the accessibility support enabled for that key (firstpluralaccess). If there is no accessibility support, this just uses \glsentryfirstplural.

\GLSaccessfmtdesc{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 519

Similar to \GLSaccessdesc but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtdesc{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 519

Similar to \Glsaccessdesc but formats the displayed text with \Glsfmtfield.

\glsaccessfmtdesc{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 519

Similar to \glsaccessdesc but formats the displayed text with \glsfmtfield.

\GLSaccessfmtdescplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 520

Similar to \GLSaccessdescplural but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtdescplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 520

Similar to \Glsaccessdescplural but formats the displayed text with \Glsfmtfield.

\glsaccessfmtdescplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 519

Similar to \glsaccessdescplural but formats the displayed text with \glsfmtfield.

\GLSaccessfmtfirst{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 518

Similar to \GLSaccessfirst but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtfirst{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 518

Similar to \Glsaccessfirst but formats the displayed text with \Glsfmtfield.

\glsaccessfmtfirst{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 518

Similar to \glsaccessfirst but formats the displayed text with \glsfmtfield.

\GLSaccessfmtfirstplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 518

Similar to \GLSaccessfirstplural but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtfirstplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 518

Similar to \Glsaccessfirstplural but formats the displayed text with \Glsfmtfield.

\glsaccessfmtfirstplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 518

Similar to \glsaccessfirstplural but formats the displayed text with \glsfmtfield.

\GLSaccessfmtlong{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \GLSaccesslong but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtlong{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \Glsaccesslong but formats the displayed text with \Glsfmtfield.

\glsaccessfmtlong{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 520

Similar to \glsaccesslong but formats the displayed text with \glsfmtfield.

\GLSaccessfmtlongpl{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \GLSaccesslongpl but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtlongpl{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \Glsaccesslongpl but formats the displayed text with \Glsfmtfield.

\glsaccessfmtlongpl{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \glsaccesslongpl but formats the displayed text with \glsfmtfield.

\GLSaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \GLSaccessname but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \Glsaccessname but formats the displayed text with \Glsfmtfield.

\glsaccessfmtname{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \glsaccessname but formats the displayed text with \glsfmtfield.

\GLSaccessfmtplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 518

Similar to \GLSaccessplural but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 518

Similar to \Glsaccessplural but formats the displayed text with \Glsfmtfield.

\glsaccessfmtplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \glsaccessplural but formats the displayed text with \glsfmtfield.

\GLSaccessfmtshort{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 520

Similar to \GLSaccessshort but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtshort{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 520

Similar to \Glsaccessshort but formats the displayed text with \Glsfmtfield.

\glsaccessfmtshort { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 520

Similar to \glsaccessshort but formats the displayed text with \glsfmtfield.

\GLSaccessfmtshortpl { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 520

Similar to \GLSaccessshortpl but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtshortpl { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 520

Similar to \Glsaccessshortpl but formats the displayed text with \Glsfmtfield.

\glsaccessfmtshortpl { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 520

Similar to \glsaccessshortpl but formats the displayed text with \glsfmtfield.

\GLSaccessfmtsymbol { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 519

Similar to \GLSaccesssymbol but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtsymbol { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 519

Similar to \Glsaccesssymbol but formats the displayed text with \Glsfmtfield.

\glsaccessfmtsymbol { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 518

Similar to \glsaccesssymbol but formats the displayed text with \glsfmtfield.

\GLSaccessfmtsymbolplural { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 519

Similar to \GLSaccesssymbolplural but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtsymbolplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 519

Similar to \Glsaccesssymbolplural but formats the displayed text with \Glsfmtfield.

\glsaccessfmtsymbolplural{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩}
glossaries−extra v1.49+ §9.3; 519

Similar to \glsaccesssymbolplural but formats the displayed text with \glsfmtfield.

\GLSaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \GLSaccesstext but formats the displayed text with \GLSfmtfield.

\Glsaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \Glsaccesstext but formats the displayed text with \Glsfmtfield.

\glsaccessfmttext{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 517

Similar to \glsaccesstext but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuseri{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \GLSaccessuseri but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuseri{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \Glsaccessuseri but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuseri{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 521

Similar to \glsaccessuseri but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuserii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \GLSaccessuserii but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuserii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \Glsaccessuserii but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuserii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 521

Similar to \glsaccessuserii but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuseriii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \GLSaccessuseriii but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuseriii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \Glsaccessuseriii but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuseriii { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \glsaccessuseriii but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuseriv { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \GLSaccessuseriv but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuseriv { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.49+ §9.3; 522

Similar to \Glsaccessuseriv but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuseriv{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 522

Similar to \glsaccessuseriv but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuserv{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 523

Similar to \GLSaccessuserv but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuserv{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 523

Similar to \Glsaccessuserv but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuserv{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 522

Similar to \glsaccessuserv but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuservi{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 523

Similar to \GLSaccessuservi but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuservi{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 523

Similar to \Glsaccessuservi but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuservi{⟨*insert*⟩}{⟨*cs*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.3; 523

Similar to \glsaccessuservi but formats the displayed text with \glsfmtfield.

\glsaccessibility[⟨*options*⟩]{⟨*PDF element*⟩}{⟨*value*⟩}{⟨*content*⟩}
glossaries−accsupp v4.45+

Applies ⟨*value*⟩ as the accessibility attribute ⟨*PDF element*⟩ for the given ⟨*content*⟩. This internally uses the accessibility support provided by accsupp.

\GLSaccesslong{⟨*entry-label*⟩} §9.2; 514

The all caps version of \glsaccesslong.

\Glsaccesslong{⟨*entry-label*⟩} §9.2; 514

The sentence case version of \glsaccesslong.

\glsaccesslong{⟨*entry-label*⟩} §9.2; 514

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the long key with the accessibility support enabled for that key (longaccess). If there is no accessibility support, this just uses \glsentrylong.

\GLSaccesslongpl{⟨*entry-label*⟩} §9.2; 514

The all caps version of \glsaccesslongpl.

\Glsaccesslongpl{⟨*entry-label*⟩} §9.2; 514

The sentence case version of \glsaccesslongpl.

\glsaccesslongpl{⟨*entry-label*⟩} §9.2; 514

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the longplural key with the accessibility support enabled for that key (longpluralaccess). If there is no accessibility support, this just uses \glsentrylongpl.

\GLSaccessname{⟨*entry-label*⟩} §9.2; 510

The all caps version of \glsaccessname.

\Glsaccessname{⟨*entry-label*⟩} §9.2; 510

The sentence case version of \glsaccessname.

\glsaccessname{⟨*entry-label*⟩} §9.2; 510

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the name key with the accessibility support enabled for that key (access). If there is no accessibility support, this just uses \glsentryname.

\GLSaccessplural{⟨*entry-label*⟩} §9.2; 511

The all caps version of \glsaccessplural.

\Glsaccessplural{⟨*entry-label*⟩} §9.2; 511

The sentence case version of \glsaccessplural.

\glsaccessplural{⟨*entry-label*⟩} §9.2; 511

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the plural key with the accessibility support enabled for that key (pluralaccess). If there is no accessibility support, this just uses \glsentryplural.

\GLSaccessshort{⟨*entry-label*⟩} §9.2; 513

The all caps version of \glsaccessshort.

\Glsaccessshort{⟨*entry-label*⟩} §9.2; 513

The sentence case version of \glsaccessshort.

\glsaccessshort{⟨*entry-label*⟩} §9.2; 513

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will

display the value of the short key with the accessibility support enabled for that key (shortaccess). If there is no accessibility support, this just uses \glsentryshort.

\GLSaccessshortpl{⟨*entry-label*⟩} §9.2; 514

The all caps version of \glsaccessshortpl.

\Glsaccessshortpl{⟨*entry-label*⟩} §9.2; 514

The sentence case version of \glsaccessshortpl.

\glsaccessshortpl{⟨*entry-label*⟩} §9.2; 513

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the shortplural key with the accessibility support enabled for that key (shortpluralaccess). If there is no accessibility support, this just uses \glsentryshortpl.

\GLSaccesssymbol{⟨*entry-label*⟩} §9.2; 512

The all caps version of \glsaccesssymbol.

\Glsaccesssymbol{⟨*entry-label*⟩} §9.2; 512

The sentence case version of \glsaccesssymbol.

\glsaccesssymbol{⟨*entry-label*⟩} §9.2; 512

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the symbol key with the accessibility support enabled for that key (symbolaccess). If there is no accessibility support, this just uses \glsentrysymbol.

\GLSaccesssymbolplural{⟨*entry-label*⟩} §9.2; 512

The all caps version of \glsaccesssymbolplural.

\Glsaccesssymbolplural{⟨*entry-label*⟩} §9.2; 512

The sentence case version of \glsaccesssymbolplural.

\glsaccesssymbolplural{⟨*entry-label*⟩} §9.2; 512

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the symbolplural key with the accessibility support enabled for that key (symbolpluralaccess). If there is no accessibility support, this just uses \glsentrysymbolplural.

\GLSaccesstext{⟨*entry-label*⟩} §9.2; 511

The all caps version of \glsaccesstext.

\Glsaccesstext{⟨*entry-label*⟩} §9.2; 511

The sentence case version of \glsaccesstext.

\glsaccesstext{⟨*entry-label*⟩} §9.2; 510

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the text key with the accessibility support enabled for that key (textaccess). If there is no accessibility support, this just uses \glsentrytext.

\GLSaccessuseri{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The all caps version of \glsaccessuseri.

\Glsaccessuseri{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The sentence case version of \glsaccessuseri.

\glsaccessuseri{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 514

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user1 key with the accessibility support enabled for that key (user1-access). If there is no accessibility support, this just uses \glsentryuseri.

\GLSaccessuserii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The all caps version of \glsaccessuserii.

\Glsaccessuserii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The sentence case version of \glsaccessuserii.

\glsaccessuserii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user2 key with the accessibility support enabled for that key (user2-access). If there is no accessibility support, this just uses \glsentryuserii.

\GLSaccessuseriii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The all caps version of \glsaccessuseriii.

\Glsaccessuseriii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

The sentence case version of \glsaccessuseriii.

\glsaccessuseriii{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user3 key with the accessibility support enabled for that key (user3-access). If there is no accessibility support, this just uses \glsentryuseriii.

\GLSaccessuseriv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The all caps version of \glsaccessuseriv.

\Glsaccessuseriv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The sentence case version of \glsaccessuseriv.

\glsaccessuseriv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 515

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user4 key with the accessibility support enabled for that key (user4-access). If there is no accessibility support, this just uses \glsentryuseriv.

\GLSaccessuserv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The all caps version of \glsaccessuserv.

\Glsaccessuserv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The sentence case version of \glsaccessuserv.

\glsaccessuserv{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user5 key with the accessibility support enabled for that key (user5-access). If there is no accessibility support, this just uses \glsentryuserv.

\GLSaccessuservi{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The all caps version of \glsaccessuservi.

\Glsaccessuservi{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

The sentence case version of \glsaccessuservi.

\glsaccessuservi{⟨*entry-label*⟩} glossaries−extra v1.49+ §9.2; 516

If accessibility support was enabled when glossaries−extra was loaded (accsupp) this will display the value of the user6 key with the accessibility support enabled for that key (user6-access). If there is no accessibility support, this just uses \glsentryuservi.

\glsaccsupp{⟨*replacement*⟩}{⟨*content*⟩} glossaries−accsupp

Applies ⟨*replacement*⟩ as the ActualText for ⟨*content*⟩ using \glsaccessibility.

\glsacspace{⟨*label*⟩} glossaries v4.16+

Uses a non-breakable space if the short form is less than \glsacspacemax otherwise uses \space. This command is provided by glossaries but has a hard-coded maximum of 3em. This command is redefined by glossaries−extra to use \glsacspacemax.

\glsacspacemax

Expands to the maximum value used by \glsacspace. This is a macro not a register. The default is 3em.

\glsadd[⟨*options*⟩]{⟨*entry-label*⟩} glossaries

Indexes the entry identified by ⟨*entry-label*⟩.

\glsaddall[⟨*options*⟩] glossaries

Iterates over all glossaries (or all those listed in the types option) and indexes each entry in the glossary. The optional argument ⟨*options*⟩ are passed to \glsadd. This command can't be used with bib2gls. Use the selection=all resource option instead.

\glsaddallunindexed[⟨*glossary types*⟩] glossaries−extra v1.49+ §5.8; 266

Iterates over all glossaries listed in ⟨*glossary types*⟩ and indexes each entry (with format=glsignore) that hasn't already been indexed. This command can't be used with bib2gls. Use the selection=all resource option instead.

\glsaddallunused[⟨*glossary types*⟩] glossaries v3.08a+

Iterates over all glossaries listed in ⟨*glossary types*⟩ and indexes each entry (with `format=glsignore`) that hasn't been used. This command can't be used with `bib2gls`. Use the `selection=all` resource option instead.

\glsaddeach[⟨*options*⟩]{⟨*entry label list*⟩} glossaries−extra v1.31+ §5.8; 266

Does `\glsadd`[⟨*options*⟩]{⟨*entry-label*⟩} for each label in the supplied comma-separated list.

\glsaddkey{⟨*key*⟩}{⟨*default value*⟩}{⟨*no link cs*⟩}{⟨*no link ucfirst cs*⟩}{⟨*link cs*⟩}{⟨*link ucfirst cs*⟩}{⟨*link allcaps cs*⟩} glossaries v3.12a

Defines a new glossary entry key with the given default value and commands that are analogous to `\glsentrytext` (⟨*no link cs*⟩), `\Glsentrytext` (⟨*no link ucfirst cs*⟩), `\glstext` (⟨*link cs*⟩), `\Glstext` (⟨*link ucfirst cs*⟩), `\GLStext` (⟨*link allcaps cs*⟩). The starred version switches on field expansion for the given key.

\glsaddpostsetkeys glossaries−extra v1.30+ §5.1.1; 192

Hook implemented after setting the options passed to `\glsadd`.

\glsaddpresetkeys glossaries−extra v1.30+ §5.1.1; 192

Hook implemented before setting the options passed to `\glsadd`.

\glsaddstoragekey{⟨*key*⟩}{⟨*default value*⟩}{⟨*no link cs*⟩} glossaries v4.16

Provides a new glossary entry key with a default value and a command for simply accessing the value (without indexing or hyperlinks). The starred version switches on field expansion for the given key.

\glsaltlistitem{⟨*entry-label*⟩} glossaries−extra−stylemods v1.47+ §8.6.5.3; 445

Used to display the top-level entry item in the altlist styles.

\glsalttreechildpredesc glossaries−extra−stylemods v1.46+ §8.6.5.4; 448

Inserted before the child descriptions for the alttree styles.

\glsalttreepredesc glossaries−extra−stylemods v1.46+ §8.6.5.4; 448

Inserted before the top-level descriptions for the alttree styles.

\glsapptopostlink { ⟨*category*⟩ } { ⟨*code*⟩ } glossaries−extra v1.49+ §5.5.4; 256

Appends ⟨*code*⟩ to post-link hook associated with the category identified by the label ⟨*category*⟩ (or simply defines it, if it doesn't already exist).

\glsautoprefix glossaries v1.14+

Expands to the prefix for the label used by `numberedsection=autolabel` and `numbered-section=nameref`.

\glsbackslash glossaries v4.11+

Expands to a literal backslash.

\glsbibdata [⟨*options*⟩] { ⟨*bib-list*⟩ } glossaries−extra v1.55+ §11; 541

Shortcut that uses `\GlsXtrLoadResources` with `src` set to ⟨*bib-list*⟩.

\glscapitalisewords { ⟨*content*⟩ } glossaries v4.48+ §5.2.4; 204

Just does `\capitalisewords` but may be redefined to use `\capitalisefmtwords`, if required.

\glscapscase { ⟨*no change*⟩ } { ⟨*sentence*⟩ } { ⟨*all caps*⟩ } glossaries

Initialised by the `\gls`-like and `\glstext`-like commands, this expands to ⟨*no change*⟩ if the calling command doesn't apply a case-change (such as `\gls` or `\glstext`), to ⟨*sentence*⟩ if

the calling command converts to sentence case (such as `\Gls` or `\Glstext`), or to ⟨*all caps*⟩ if the calling command converts to all caps (such as `\GLS` or `\GLStext`). This command may be used within associated hooks, entry display styles (`\defglsentryfmt`), and the post-link hook.

`\glscapturedgroup`⟨*n*⟩ glossaries−extra−bib2gls v1.31+ §11.6.2; 581

Expands to `\string\$`⟨*n*⟩. Note that this isn't the same as `\MGP`.

`\glscategory`{ ⟨*entry-label*⟩ } §10; 524

Expands to the entry's category.

`\glscategorylabel` §4.5.3.1; 169

Expands to the category label of the abbreviation that is in the process of being defined by `\newabbreviation`. Maybe used in the style hooks (but take care to expand this command, if necessary).

`\glscombinedfirstsep`{ ⟨*prev label*⟩ } { ⟨*next label*⟩ } glossaries−extra v1.48+ §7.4; 354

Separator used between elements of a multi-entry set where only the next element have been marked as used.

`\glscombinedfirstsepfirst`{ ⟨*prev label*⟩ } { ⟨*next label*⟩ } glossaries−extra v1.48+ §7.4; 354

Separator used between elements of a multi-entry set where neither the previous nor the next element has been marked as used.

`\glscombinedsep`{ ⟨*prev label*⟩ } { ⟨*next label*⟩ } glossaries−extra v1.48+ §7.4; 353

Separator used between elements of a multi-entry set where both elements have been marked as used.

\glscombinedsepfirst { ⟨*prev label*⟩ } { ⟨*next label*⟩ } glossaries−extra v1.48+ §7.4; 354

Separator used between elements of a multi-entry set where only the previous element have been marked as used.

\glscurrententrylabel glossaries v3.02+

Assigned at the start of each entry item within the glossary. This command may be used by glossary hooks, such as the post-description hook, to reference the current entry.

\glscurrententrylevel glossaries−extra v1.44+ §8.4.3; 413

Defined within the "unsrt" family of commands to the current hierarchical level (taking `level-offset` into account).

\glscurrentfieldvalue glossaries v4.23+

Conditional commands such as `\ifglshasfield` set this to the field's value for use within the ⟨*true*⟩ code.

\glscurrentrootentry glossaries−extra v1.49+ §8.4.3; 414

May be used within `\printunsrtglossaryentryprocesshook` to reference the most recent top level entry label (allowing for `flatten` but not `leveloffset`).

\glscurrenttoplevelentry glossaries−extra v1.49+ §8.4.3; 414

May be used within `\printunsrtglossaryentryprocesshook` to reference the most recent top level entry label (allowing for `flatten` and `leveloffset`).

\glscustomtext glossaries

The custom text provided by `\glsdisp` or the link text for the `\glstext`-like commands. This command may be used within associated hooks, entry display styles (`\defglsentryfmt`), and the post-link hook.

\glsdefaultshortaccess{⟨*long*⟩}{⟨*short*⟩} glossaries−accsupp v4.45+
(requires accsupp)

Used when \newabbreviation automatically assigns shortaccess. This is defined by glossaries−accsupp to just do ⟨*long*⟩ but is redefined by glossaries−extra to do ⟨*long*⟩ (⟨*short*⟩).

\glsdefaulttype *initial:* main glossaries

Expands to the label of the default glossary, which is normally main but if nomain is used, it will be the label of the first glossary to be defined.

\glsdefpostdesc{⟨*category*⟩}{⟨*definition*⟩} glossaries−extra v1.31+ §8.6.2; 439

Defines post-description hook associated with the category identified by the label ⟨*category*⟩. This simply (re)defines \glsxtrpostdesc⟨*category*⟩ for the given ⟨*category*⟩ to ⟨*definition*⟩.

\glsdefpostlink{⟨*category*⟩}{⟨*definition*⟩} glossaries−extra v1.31+ §5.5.4; 255

Defines post-link hook associated with the category identified by the label ⟨*category*⟩. This simply (re)defines \glsxtrpostlink⟨*category*⟩ for the given ⟨*category*⟩ to ⟨*definition*⟩.

\glsdefpostname{⟨*category*⟩}{⟨*definition*⟩} glossaries−extra v1.31+ §8.6.1; 437

Defines post-name hook associated with the category identified by the label ⟨*category*⟩. This simply (re)defines \glsxtrpostname⟨*category*⟩ for the given ⟨*category*⟩ to ⟨*definition*⟩.

\GLSdesc[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsdesc but converts the link text to all caps.

\Glsdesc [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsdesc but converts the link text to sentence case. Use \Glossentrydesc within custom glossary styles instead of this command.

\glsdesc [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the description value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options. Use \glossentrydesc within custom glossary styles instead of this command.

\glsdescplural [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries

As \glsdesc but for the descriptionplural field.

\glsdescriptionaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ }
glossaries-accsupp

Does ⟨*text*⟩ with the descriptionaccess replacement text (if set).

\glsdescriptionpluralaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ }
glossaries-accsupp

Does ⟨*text*⟩ with the descriptionpluralaccess replacement text (if set).

\glsdescwidth glossary-long & glossary-super

A length register used to set the width of the description column for tabular-like styles.

\Glsdisp [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*text*⟩ } *modifiers:* * + ⟨*alt-mod*⟩
glossaries v4.50+

As \glsdisp but sets the link text to \glssentencecase{⟨*text*⟩}. This is provided to allow a sentence case mapping in the event that \glsdisp occurs at the start of content that has automated case-changing.

\glsdisp [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*text*⟩ } *modifiers:* * + ⟨*alt-mod*⟩
glossaries v1.19+

References the entry identified by ⟨*entry-label*⟩ with the given ⟨*text*⟩ as the link text. This command unsets the first use flag (use \glslink instead, if the first use flag should not be altered). This command is considered a \gls-like command. For the first optional argument, see \glslink options.

\glsdisplaynumberlist { ⟨*entry-label*⟩ } glossaries v3.02+

§11.6.8; 621

Formats the location list for the given entry. Redefined by glossaries−extra−bib2gls to obtain the location list from the location field.

\glsdoifexists { ⟨*entry-label*⟩ } { ⟨*code*⟩ } glossaries

Does ⟨*code*⟩ if the entry given by ⟨*entry-label*⟩ exists. If the entry doesn't exist, this will either generate an error (undefaction=error) or a warning (undefaction=warn) and, within the document environment, it will insert the unknown marker ??.

\glsdoifexistsordo { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v4.19+

Similar to \ifglsentryexists, this does ⟨*true*⟩ if the entry given by ⟨*entry-label*⟩ exists. If the entry doesn't it exist, this does ⟨*false*⟩ and generates an error (undefaction=error) or a warning (undefaction=warn). The unknown marker ?? will be placed before the ⟨*false*⟩ code.

\glsdoifexistsorwarn { ⟨*entry-label*⟩ } { ⟨*code*⟩ } glossaries v4.03+

Like \glsdoifexists, but always warns (no error) if the entry doesn't exist, regardless of the undefaction setting, and doesn't show the unknown marker.

\glsdoifnoexists { ⟨*entry-label*⟩ } { ⟨*code*⟩ } glossaries

Does ⟨*code*⟩ if the entry given by ⟨*entry-label*⟩ does not exist. If the entry does exist, this will either generate an error (`undefaction=error`) or a warning (`undefaction=warn`).

\glsenableentrycount glossaries v4.14+

Enables entry counting.

\glsenableentryunitcount §6.1; 323

Enables entry unit counting.

\glsencapwrcontent { ⟨*code*⟩ } glossaries v4.50+ & glossaries−extra v1.49+ §5.8; 272

Encapsulates the indexing code (within `\glslinkwrcontent`).

\glsendrange [⟨*options*⟩] { ⟨*entry label list*⟩ } glossaries−extra v1.50+ §5.8; 267

As `\glsstartrange` but with the end range marker) .

\glsentrycounterlabel glossaries v3.0+

Used by `\glsentryitem` to display the entry counter label.

\glsentrycurrcount { ⟨*entry-label*⟩ } glossaries v4.14+

Expands to the current entry count running total or 0 if not available (needs to be enabled with `\glsenableentrycount` or `\glsenableentryunitcount`). With unit entry counting, this expands to the total for the current unit.

\Glsentrydesc { ⟨*entry-label*⟩ } glossaries

Partially robust command that displays the value of the `description` field with sentence

case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrydesc{⟨*entry-label*⟩} glossaries

Simply expands to the value of the description field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the description field doesn't contain any fragile commands.

\glsentrydescplural{⟨*entry-label*⟩} glossaries

Simply expands to the value of the descriptionplural field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the description-plural field doesn't contain any fragile commands.

\Glsentryfirst{⟨*entry-label*⟩} glossaries

Partially robust command that displays the value of the first field with the first letter converted to uppercase. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryfirst{⟨*entry-label*⟩} glossaries

Simply expands to the value of the first field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the first field doesn't contain any fragile commands.

\Glsentryfirstplural{⟨*entry-label*⟩} glossaries

Partially robust command that displays the value of the firstplural field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryfirstplural{⟨*entry-label*⟩} glossaries

Simply expands to the value of the firstplural field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the firstplural field doesn't contain any fragile commands.

\glsentryfmt glossaries v3.11a+

The default display format used by the \gls-like commands. This checks if the short field has been set for the current entry and, if set, initialises the abbreviation formatting commands (with \glssetabbrvfmt). This command will do \glsgenentryfmt (encapsulated with \glsxtrregularfont) if the entry is considered a regular entry (\glsifregular) or if the entry doesn't have the short field set. Otherwise it will do \glsxtrgenabbrvfmt encapsulated with \glsxtrabbreviationfont.

\glsentryindexcount{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.8; 272

Expands to the number of times the given entry has been indexed. This will expand to 0 if the entry hasn't been indexed or hasn't been defined.

\glsentryitem{⟨*entry-label*⟩} glossaries v3.0+

Does nothing if entrycounter=false, otherwise increments and displays the associated counter.

\Glsentrylong{⟨*entry-label*⟩} glossaries v3.0+

Displays the value of the long field with sentence case applied. Does nothing if the entry hasn't been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrylong{⟨*entry-label*⟩} glossaries v3.0+

Simply expands to the value of the long field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the long field doesn't contain any fragile commands.

\Glsentrylongpl{⟨*entry-label*⟩} glossaries v3.0+

Displays the value of the longplural field with sentence case applied. Does nothing if the entry hasn't been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrylongpl{⟨*entry-label*⟩} glossaries v3.0+

Simply expands to the value of the `longplural` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `longplural` field doesn't contain any fragile commands.

\Glsentryname{⟨*entry-label*⟩} glossaries

Partially robust command that displays the value of the `name` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryname{⟨*entry-label*⟩} glossaries

Simply expands to the value of the `name` key. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `name` key doesn't contain any fragile commands.

\glsentrynumberlist{⟨*entry-label*⟩} glossaries v3.02+

§11.6.8; 621

Displays the location list for the given entry. Redefined by glossaries−extra−bib2gls to obtain the location list from the `location` field.

\glsentryparent{⟨*entry-label*⟩} glossaries

Expands to the value of the `parent` field. Expands to nothing if the `parent` field hasn't been set and expands to `\relax` if the entry hasn't been defined.

\glsentrypdfsymbol{⟨*entry-label*⟩} glossaries−extra v1.42+

§8.6; 435

Used when `\glossentrysymbol` occurs in a PDF bookmark.

\Glsentryplural{⟨*entry-label*⟩} glossaries

Partially robust command that displays the value of the `plural` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF

bookmarks it will expand to a robust internal command.

\glsentryplural{⟨*entry-label*⟩} glossaries

Simply expands to the value of the `plural` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `plural` field doesn't contain any fragile commands.

\glsentryprevcount{⟨*entry-label*⟩} glossaries v4.14+

Expands to the final entry count total from the previous LaTeX run or if 0 if not available (needs to be enabled with `\glsenableentrycount` or `\glsenableentryunitcount`). With unit entry counting, this expands to the total for the current unit.

\glsentryprevmaxcount{⟨*entry-label*⟩} glossaries v4.14+ §6.1; 324

Expands to the maximum entry unit count total from the previous LaTeX run or if 0 if not available (needs to be enabled with `\glsenableentryunitcount`).

\glsentryprevtotalcount{⟨*entry-label*⟩} glossaries v4.14+ §6.1; 324

Expands to the final entry count total from the previous LaTeX run or if 0 if not available (needs to be enabled with `\glsenableentryunitcount`).

\Glsentryshort{⟨*entry-label*⟩} glossaries v3.0+

Displays the value of the `short` field with sentence case applied. Does nothing if the entry hasn't been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryshort{⟨*entry-label*⟩} glossaries v3.0+

Simply expands to the value of the `short` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `short` field doesn't contain any fragile commands.

\Glsentryshortpl{⟨*entry-label*⟩} glossaries v3.0+

Displays the value of the shortplural field with sentence case applied. Does nothing if the entry hasn't been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryshortpl{⟨*entry-label*⟩} glossaries v3.0+

Simply expands to the value of the shortplural field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the shortplural field doesn't contain any fragile commands.

\Glsentrysymbol{⟨*entry-label*⟩} glossaries

Partially robust command that displays the value of the symbol field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrysymbol{⟨*entry-label*⟩} glossaries

Simply expands to the value of the symbol field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the symbol field doesn't contain any fragile commands.

\glsentrysymbolaccess{⟨*entry-label*⟩} glossaries−accsupp

As \glsentrysymbol but for the symbolaccess field.

\glsentrysymbolplural{⟨*entry-label*⟩} glossaries

Simply expands to the value of the symbolplural field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the symbolplural field doesn't contain any fragile commands.

\Glsentrytext { ⟨*entry-label*⟩ } glossaries

Partially robust command that displays the value of the `text` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrytext { ⟨*entry-label*⟩ } glossaries

Simply expands to the value of the `text` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `text` field doesn't contain any fragile commands.

\glsentrytitlecase { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } glossaries v4.22+

Applies title case to the value supplied in the given field (which is obtained with `\glsfieldfetch`).

\glsentrytype { ⟨*entry-label*⟩ } glossaries

Simply expands to the value of the `type` key. Does nothing if the entry hasn't been defined.

\Glsentryuseri { ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user1` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseri { ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user1` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user1` field doesn't contain any fragile commands.

\Glsentryuserii { ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user2` field with sentence case ap-

plied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuserii{ ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user2` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user2` field doesn't contain any fragile commands.

\Glsentryuseriii{ ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user3` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseriii{ ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user3` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user3` field doesn't contain any fragile commands.

\Glsentryuseriv{ ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user4` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseriv{ ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user4` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user4` field doesn't contain any fragile commands.

\Glsentryuserv{ ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user5` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuserv{ ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user5` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user5` field doesn't contain any fragile commands.

\Glsentryuservi{ ⟨*entry-label*⟩ } glossaries v2.04+

Partially robust command that displays the value of the `user6` field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuservi{ ⟨*entry-label*⟩ } glossaries v2.04+

Simply expands to the value of the `user6` field. Does nothing if the entry hasn't been defined. May be used in expandable contexts provided that the `user6` field doesn't contain any fragile commands.

\glsexclapplyinnerfmtfield{ ⟨*entry-label*⟩ } { ⟨*internal-field*⟩ }
glossaries−extra v1.49+

§5.5.3; 250

Locally adds the field given by its internal field label ⟨*internal-field*⟩ to the inner formatting exclusion list for the entry identified by ⟨*entry-label*⟩. This typically means that the field value already contains the inner formatting.

\glsexpandfields glossaries v3.08a+

Expand field values when defining entries, except for those that explicitly have expansion disabled with `\glssetnoexpandfield`.

\glsextrapostnamehook{ ⟨*entry-label*⟩ } glossaries−extra v1.25+

§8.6.1; 437

A general purpose hook that's performed within `\glsxtrpostnamehook`.

\glsfieldaccsupp{ ⟨*replacement*⟩ } { ⟨*content*⟩ } { ⟨*field*⟩ } { ⟨*entry-label*⟩ }
glossaries−accsupp v4.45+

If glossaries−extra has been loaded, this command will first check for the existence of the command \glsxtr⟨*category*⟩⟨*field*⟩accsupp. If that command doesn't exist or if glossaries−extra hasn't been loaded, it then checks for the existence of \gls⟨*field*⟩accsupp (for example, \glsshortaccsupp). Failing that it will use \glsaccsupp. Whichever command is found first, ⟨*cs*⟩ { ⟨*replacement*⟩ } { ⟨*content*⟩ } is performed.

\glsfielddef{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ } glossaries v4.16+

Locally assigns the ⟨*value*⟩ to the given field (identified by the internal field label ⟨*field*⟩) for the entry identified by ⟨*entry-label*⟩. Produces an error (or warning with undefaction=warn) if the entry or field doesn't exist. Note that this doesn't update any associated fields.

\glsfieldedef{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ } glossaries v4.16+

Locally assigns the full expansion of ⟨*value*⟩ to the given field (identified by the internal field label ⟨*field*⟩) for the entry identified by ⟨*entry-label*⟩. Produces an error (or warning with undefaction=warn) if the entry or field doesn't exist. Note that this doesn't update any associated fields.

\glsfieldfetch{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*cs*⟩ } glossaries v4.16+

Fetches the value of the given field for the given entry and stores it in the command ⟨*cs*⟩. Triggers an error if the given field (identified by its internal field label) hasn't been defined. Uses \glsdoifexists.

\glsfieldgdef{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ } glossaries v4.16+

As \glsfielddef but does a global assignment.

\glsfieldxdef{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ } glossaries v4.16+

As \glsfieldedef but does a global assignment.

\glsFindWidestAnyName [⟨*glossary labels*⟩] glossaries−extra−stylemods v1.05+ §8.6.5.4; 451

Finds and sets the widest name for all entries in the given glossaries.

\glsFindWidestAnyNameLocation [⟨*glossary labels*⟩] { ⟨*register*⟩ }
glossaries−extra−stylemods v1.05+ §8.6.5.4; 452

Like `\glsFindWidestAnyName` but also also measures the location list. The length of the widest location is stored in ⟨*register*⟩, which should be a length register.

\glsFindWidestAnyNameSymbol [⟨*glossary labels*⟩] { ⟨*register*⟩ }
glossaries−extra−stylemods v1.05+ §8.6.5.4; 452

Like `\glsFindWidestAnyName` but also also measures the symbol. The length of the widest symbol is stored in ⟨*register*⟩ which should be a length register.

\glsFindWidestAnyNameSymbolLocation [⟨*glossary labels*⟩] { ⟨*register1*⟩ } { ⟨*register2*⟩ } glossaries−extra−stylemods v1.05+ §8.6.5.4; 452

Like `\glsFindWidestAnyNameSymbol` but also also measures the location list. The length of the widest symbol is stored in ⟨*register1*⟩ and the length of the widest location is stored in ⟨*register2*⟩, which should both be length registers.

\glsFindWidestLevelTwo [⟨*glossary labels*⟩] glossaries−extra−stylemods v1.05+ §8.6.5.4; 451

Finds and sets the widest name for all entries with hierarchical level less than or equal to 2 in the given glossaries.

\glsFindWidestTopLevelName glossaries−extra−stylemods §8.6.5.4; 451

A synonym for `\glsfindwidesttoplevelname`.

\glsfindwidesttoplevelname [⟨*glossary labels*⟩] glossary−tree v4.22+

Finds and sets the widest name for all top-level entries in the given glossaries. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\glsFindWidestUsedAnyName [⟨*glossary labels*⟩]
glossaries−extra−stylemods v1.05+

§8.6.5.4; 451

Finds and sets the widest name for all entries that have been marked as used in the given glossaries.

\glsFindWidestUsedAnyNameLocation [⟨*glossary labels*⟩] { ⟨*register*⟩ }
glossaries−extra−stylemods v1.05+

§8.6.5.4; 452

Like `\glsFindWidestUsedAnyName` but also also measures the location list. The length of the widest location is stored in ⟨*register*⟩, which should be a length register.

\glsFindWidestUsedAnyNameSymbol [⟨*glossary labels*⟩] { ⟨*register*⟩ }
glossaries−extra−stylemods v1.05+

§8.6.5.4; 451

Like `\glsFindWidestUsedAnyName` but also also measures the symbol. The length of the widest symbol is stored in ⟨*register*⟩ which should be a length register.

\glsFindWidestUsedAnyNameSymbolLocation [⟨*glossary labels*⟩] { ⟨*register1*⟩ } { ⟨*register2*⟩ }
glossaries−extra−stylemods v1.05+

§8.6.5.4; 452

Like `\glsFindWidestUsedAnyNameSymbol` but also also measures the location list. The length of the widest symbol is stored in ⟨*register1*⟩ and the length of the widest location is stored in ⟨*register2*⟩, which should both be length registers.

\glsFindWidestUsedLevelTwo [⟨*glossary labels*⟩]
glossaries−extra−stylemods v1.05+

§8.6.5.4; 451

Finds and sets the widest name for all entries that have been marked as used with hierarchical level less than or equal to 2 in the given glossaries.

\glsFindWidestUsedTopLevelName [⟨*glossary labels*⟩]
glossaries−extra−stylemods v1.05+

§8.6.5.4; 451

Finds and sets the widest name for all top-level entries that have been marked as used in the given glossaries.

\GLSfirst [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsfirst but converts the link text to all caps. If you have defined the entry with \newabbreviation use \GLSxtrfull or \GLS[prereset] instead.

\Glsfirst [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsfirst but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc. If you have defined the entry with \newabbreviation use \Glsxtrfull or \Gls[prereset] instead.

\glsfirst [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the first value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \newabbreviation use \glsxtrfull for the full form or \glsxtrlong for the long form or use \gls[prereset], as some abbreviation styles are too complicated to work with \glsfirst. For the first optional argument, see \glslink options.

\glsfirstabbrvdefaultfont {⟨*text*⟩} 139

Formatting command for the short form on first use used by the abbreviation styles that don't apply a font change by default.

\glsfirstabbrvemfont {⟨*text*⟩} glossaries−extra v1.04+ 163

Short form font used by the "em" abbreviation styles on first use.

\glsfirstabbrvfont {⟨*text*⟩} 176

Font formatting command for the short form on first use, initialised by the abbreviation style.

\glsfirstabbrvhyphenfont{⟨*text*⟩} glossaries−extra v1.17+ 154

Short form font used by the "hyphen" abbreviation styles on first use.

\glsfirstabbrvonlyfont{⟨*text*⟩} glossaries−extra v1.17+ 160

Short form font used by the "only" abbreviation styles on first use.

\glsfirstabbrvscfont{⟨*text*⟩} glossaries−extra v1.17+ 162

Short form font used by the small caps "sc" abbreviation styles on first use.

\glsfirstabbrvsconlyfont{⟨*text*⟩} glossaries−extra v1.48+ 160

Short form font used by the "sc-only" abbreviation styles on first use.

\glsfirstabbrvscuserfont{⟨*text*⟩} glossaries−extra v1.48+ 143

Short form font used by the small caps "sc-user" abbreviation styles on first use.

\glsfirstabbrvsmfont{⟨*text*⟩} glossaries−extra v1.17+ 163

Short form font used by the "sm" abbreviation styles on first use.

\glsfirstabbrvuserfont{⟨*text*⟩} glossaries−extra v1.04+ 142

Short form font used by the "user" abbreviation styles on first use.

\glsfirstaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−accsupp

Does ⟨*text*⟩ with the `firstaccess` replacement text (if set).

\glsfirstinnerfmtabbrvfont{⟨*text*⟩} glossaries−extra v1.49+ §4.5.3.1; 172

Applies both `\glsfirstabbrvfont` and `\glsxtrgenentrytextfmt` to ⟨*text*⟩.

\glsfirstinnerfmtlongfont{⟨*text*⟩} glossaries−extra v1.49+ §4.5.3.1; 173

Applies both `\glsfirstlongfont` and `\glsxtrgenentrytextfmt` to ⟨*text*⟩.

\glsfirstlongdefaultfont{⟨*text*⟩} 140

Formatting command for the long form on first use used by the abbreviation styles that don't apply a font change by default.

\glsfirstlongemfont{⟨*text*⟩} glossaries−extra v1.04+ 164

Long form font used by the "em" abbreviation styles on first use.

\glsfirstlongfont{⟨*text*⟩} 177

Font formatting command for the long form on first use, initialised by the abbreviation style.

\glsfirstlongfootnotefont{⟨*text*⟩} glossaries−extra v1.05+ 151

Formatting command for the first use long form used by the footnote abbreviation styles.

\glsfirstlonghyphenfont{⟨*text*⟩} glossaries−extra v1.17+ 154

Long form font used by the "hyphen" abbreviation styles on first use.

\glsfirstlongonlyfont{⟨*text*⟩} glossaries−extra v1.17+ 160

Long form font used by the "only" abbreviation styles on first use.

\glsfirstlonguserfont{⟨*text*⟩} glossaries−extra v1.04+ 143

Long form font used by the "user" abbreviation styles on first use.

\GLSfirstplural[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries

As \glsfirstplural but converts the link text to all caps. If you have defined the entry with \newabbreviation use \GLSxtrfullpl or \GLSpl[prereset] instead.

\Glsfirstplural[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries

As \glsfirstplural but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc. If you have defined the entry with \newabbreviation use \Glsxtrfullpl or \Glspl[prereset] instead.

\glsfirstplural[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the firstplural value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \newabbreviation use \glsxtrfullpl for the full form or \glsxtrlongpl for the long form or use \glspl[prereset], as some abbreviation styles are too complicated to work with \glsfirstplural. For the first optional argument, see \glslink options.

\glsfirstpluralaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩}
glossaries−accsupp

Does ⟨*text*⟩ with the firstpluralaccess replacement text (if set).

\glsfirstxpabbrvfont{⟨*text*⟩}{⟨*category*⟩} glossaries−extra v1.49+ §4.5.3.1; 172

If the markshortwords attribute is set for the given category, this encapsulates ⟨*text*⟩

with \glsfirstabbrvfont otherwise with \glsfirstinnerfmtabbrvfont. This command has to expand, so protect any content that shouldn't expand.

\glsfirstxplongfont { ⟨*category*⟩ } { ⟨*text*⟩ } glossaries−extra v1.49+ §4.5.3.1; 173

If the markwords attribute is set for the given category, this encapsulates ⟨*text*⟩ with \glsfirstlongfont otherwise with \glsinnerfmtlongfont. This command has to expand, so protect any content that shouldn't expand.

\GLSfmtfield { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } { ⟨*internal-field*⟩ }
glossaries−extra v1.49+ §5.5.3; 250

As \glsfmtfield but changes the field value to all caps.

\Glsfmtfield { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } { ⟨*internal-field*⟩ }
glossaries−extra v1.49+ §5.5.3; 250

As \glsfmtfield but uses \makefirstuc to change the field value to sentence case.

\glsfmtfield { ⟨*insert*⟩ } { ⟨*cs*⟩ } { ⟨*entry-label*⟩ } { ⟨*internal-field*⟩ }
glossaries−extra v1.49+ §5.5.3; 250

Applies the formatting command ⟨*cs*⟩ (which takes one argument) to the entry's field value identified by the given internal field label, including ⟨*insert*⟩ appended. Used by the inner formatting commands. Note that \glsfmtfield should not be robust as it needs to expand if it's inside a case-changing command.

\GLSfmtfirst { ⟨*entry-label*⟩ } glossaries−extra v1.42+ §5.3.2; 217

For use within captions or section titles to display the formatted all caps first.

\Glsfmtfirst { ⟨*entry-label*⟩ } §5.3.2; 217

For use within captions or section titles to display the formatted sentence case first.

\glsfmtfirst{⟨*entry-label*⟩} §5.3.2; 217

For use within captions or section titles to display the formatted `first`.

\GLSfmtfirstpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 217

For use within captions or section titles to display the formatted all caps `firstplural`.

\Glsfmtfirstpl{⟨*entry-label*⟩} §5.3.2; 217

For use within captions or section titles to display the formatted sentence case `firstplural`.

\glsfmtfirstpl{⟨*entry-label*⟩} §5.3.2; 217

For use within captions or section titles to display the formatted `firstplural`.

\GLSfmtfull{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 215

Designed for use in section headings or captions, this expands to just `\glspdffmtfull`{⟨*entry-label*⟩} in PDF bookmarks (no case-change), otherwise it expands to `\GLSxtrtitlefull`{⟨*entry-label*⟩}.

\Glsfmtfull{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.2; 214

Designed for use in section headings or captions, this expands to just `\glspdffmtfull`{⟨*entry-label*⟩} in PDF bookmarks (no case-change), otherwise it expands to `\Glsxtrtitlefull`{⟨*entry-label*⟩}.

\glsfmtfull{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.2; 214

Designed for use in section headings or captions, this expands to just `\glspdffmtfull`{⟨*entry-label*⟩} in PDF bookmarks, otherwise it expands to `\glsxtrtitlefull`{⟨*entry-label*⟩}.

\GLSfmtfullpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 215

Designed for use in section headings or captions, this expands to just \glspdffmtfullpl{⟨*entry-label*⟩} in PDF bookmarks (no case-change), otherwise it expands to \GLSxtrtitlefullpl{⟨*entry-label*⟩}.

\Glsfmtfullpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.2; 215

Designed for use in section headings or captions, this expands to just \glspdffmtfullpl{⟨*entry-label*⟩} in PDF bookmarks (no case-change), otherwise it expands to \Glsxtrtitlefullpl{⟨*entry-label*⟩}.

\glsfmtfullpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.2; 215

Designed for use in section headings or captions, this expands to just \glspdffmtfullpl{⟨*entry-label*⟩} in PDF bookmarks, otherwise it expands to \glsxtrtitlefullpl{⟨*entry-label*⟩}.

\GLSfmtinsert glossaries−extra v1.49+

As \glsfmtinsert but converts to all caps.

\glsfmtinsert glossaries−extra v1.49+

A shortcut that applies \glsxtrgenentrytextfmt to \glsinsert if \glsinsert isn't empty.

\GLSfmtlong{⟨*entry-label*⟩} §5.3.2; 213

For use within captions or section titles to display the formatted all caps long form.

\Glsfmtlong{⟨*entry-label*⟩} §5.3.2; 212

For use within captions or section titles to display the formatted sentence case long form.

\glsfmtlong{⟨*entry-label*⟩} §5.3.2; 212

For use within captions or section titles to display the formatted long form.

\GLSfmtlongpl{⟨*entry-label*⟩} §5.3.2; 213

For use within captions or section titles to display the formatted all caps long plural form.

\Glsfmtlongpl{⟨*entry-label*⟩} §5.3.2; 213

For use within captions or section titles to display the formatted sentence case long plural form.

\glsfmtlongpl{⟨*entry-label*⟩} §5.3.2; 213

For use within captions or section titles to display the formatted long plural form.

\GLSfmtname{⟨*entry-label*⟩} glossaries−extra v1.21+ §5.3.2; 216

For use within captions or section titles to display the formatted all caps `name`.

\Glsfmtname{⟨*entry-label*⟩} glossaries−extra v1.21+ §5.3.2; 215

For use within captions or section titles to display the formatted sentence case `name`.

\glsfmtname{⟨*entry-label*⟩} glossaries−extra v1.21+ §5.3.2; 215

For use within captions or section titles to display the formatted `name`.

\GLSfmtplural{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 216

For use within captions or section titles to display the formatted all caps `plural`.

\Glsfmtplural{⟨*entry-label*⟩} §5.3.2; 216

For use within captions or section titles to display the formatted sentence case `plural`.

\glsfmtplural{⟨*entry-label*⟩} §5.3.2; 216

For use within captions or section titles to display the formatted `plural`.

\GLSfmtshort{⟨*entry-label*⟩} §5.3.2; 211

For use within captions or section titles to display the formatted all caps short form.

\Glsfmtshort{⟨*entry-label*⟩} §5.3.2; 211

For use within captions or section titles to display the formatted sentence case short form.

\glsfmtshort{⟨*entry-label*⟩} §5.3.2; 211

For use within captions or section titles to display the formatted short form.

\GLSfmtshortpl{⟨*entry-label*⟩} §5.3.2; 212

For use within captions or section titles to display the formatted all caps short plural form.

\Glsfmtshortpl{⟨*entry-label*⟩} §5.3.2; 211

For use within captions or section titles to display the formatted sentence case short plural form.

\glsfmtshortpl{⟨*entry-label*⟩} §5.3.2; 211

For use within captions or section titles to display the formatted short plural form.

\GLSfmttext { ⟨*entry-label*⟩ } glossaries−extra v1.42+ §5.3.2; 216

For use within captions or section titles to display the formatted all caps `text`.

\Glsfmttext { ⟨*entry-label*⟩ } §5.3.2; 216

For use within captions or section titles to display the formatted sentence case `text`.

\glsfmttext { ⟨*entry-label*⟩ } §5.3.2; 216

For use within captions or section titles to display the formatted `text`.

\glsforeachincategory [⟨*glossary-types*⟩] { ⟨*category*⟩ } { ⟨*glossary-cs*⟩ } { ⟨*label-cs*⟩ } { ⟨*body*⟩ } §10; 525

Iterates over all entry in the given list of glossaries (or all non-ignored glossaries, if the optional argument is omitted) and performs ⟨*body*⟩ for those entries that have the `category` set to ⟨*category*⟩. Within ⟨*body*⟩, the current entry can be referenced with ⟨*label-cs*⟩ and the glossary can be referenced with ⟨*glossary-cs*⟩.

\glsforeachwithattribute [⟨*glossary-types*⟩] { ⟨*attribute-label*⟩ } { ⟨*attribute-value*⟩ } { ⟨*glossary-cs*⟩ } { ⟨*label-cs*⟩ } { ⟨*body*⟩ } §10; 525

Iterates over all entry in the given list of glossaries (or all non-ignored glossaries, if the optional argument is omitted) and performs ⟨*body*⟩ for those entries that have the attribute given by ⟨*attribute-label*⟩ set to ⟨*attribute-value*⟩. Within ⟨*body*⟩, the current entry can be referenced with ⟨*label-cs*⟩ and the glossary can be referenced with ⟨*glossary-cs*⟩.

\glsgenentryfmt glossaries v3.11a+ §5.5.5; 260

The display format used by `\glsentryfmt` for regular entries.

\glsgetattribute { ⟨*entry-label*⟩ } { ⟨*attribute*⟩ } §10.2.2; 536

Expands to the value of the given attribute for the category associated with the entry identified by ⟨*entry-label*⟩. Expands to nothing if the attribute hasn't been set.

\glsgetcategoryattribute{⟨*category*⟩}{⟨*attribute*⟩} §10.2.2; 536

Expands to the value of the given attribute for the given category. Expands to nothing if the attribute hasn't been set.

\glsgetwidestname glossaries-extra-stylemods v1.05+ §8.6.5.4; 450

Expands to the widest top-level name.

\glsgetwidestsubname{⟨*level*⟩} glossaries-extra-stylemods v1.05+ §8.6.5.4; 450

Expands to the widest name for the given hierarchical level or to the widest top-level name, if no widest name set for ⟨*level*⟩.

\glsgroupheading{⟨*group-label*⟩} glossaries

Inserted at the start of each group in a glossary (unless `groups=false`) to display the group's heading, if applicable, using the title associated with ⟨*group-label*⟩ or, if no title provided, just ⟨*group-label*⟩. This command is defined by glossary styles as appropriate.

\glsgroupskip glossaries

Inserted before each group heading (except the first) in a glossary (unless `groups=false`). This command is defined by glossary styles as appropriate. Most of the predefined styles define this command to check the `nogroupskip` option.

\glshasattribute{⟨*entry-label*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩} §10.2.2; 537

Tests if the given attribute has been set for the category associated with the entry identified by ⟨*entry-label*⟩ (using etoolbox's `\ifcsvoid`). Does ⟨*false*⟩ if the entry hasn't been defined.

\glshascategoryattribute{⟨*category*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩} §10.2.2; 537

Tests if the given attribute has been set for the given category (using etoolbox's `\ifcsvoid`).

\glshashchar glossaries−extra−bib2gls v1.49+ §11.6.2; 581

Expands to a literal hash #.

\glshex⟨*hex*⟩ glossaries−extra−bib2gls v1.21+ §11.6.2; 581

Expands to \string\u⟨*hex*⟩.

\glshyperlink[⟨*text*⟩]{⟨*entry-label*⟩} glossaries v1.17+

Creates a hyperlink to the given entry with the hyperlink text provided in the optional argument. If omitted, the default is \glsentrytext{⟨*entry-label*⟩}.

\glshypernumber{⟨*location(s)*⟩} glossaries

This will encapsulate each location with a hyperlink, if supported. This may be used as a location encap. The argument may be a single location or locations delimited by \delimR or \delimN. This command should not be used outside of location lists as it requires additional information in order to correctly form the hyperlinks.

\glsifapplyinnerfmtfield{⟨*entry-label*⟩}{⟨*internal-field*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §5.5.3; 250

Tests if the field given by its internal field label ⟨*internal-field*⟩ has been added to the inner formatting exclusion list for the entry identified by ⟨*entry-label*⟩.

\glsifattribute{⟨*entry-label*⟩}{⟨*attribute*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩} §10.2.2; 537

Tests if the category associated with the entry identified by ⟨*entry-label*⟩ has the given attribute set to ⟨*value*⟩. Does ⟨*true*⟩ if the attribute is ⟨*value*⟩ and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category or if the entry hasn't been defined.

\glsifattributetrue{⟨*entry-label*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §10.2.2; 538

Tests if the category associated with the entry given by ⟨*entry-label*⟩ has the given attribute set

to `true`. Does ⟨*true*⟩ if the attribute is `true` and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category or if the entry hasn't been defined.

\glsifcategory{⟨*entry-label*⟩}{⟨*category*⟩}{⟨*true*⟩}{⟨*false*⟩} §10; 524

Tests if the entry identified by ⟨*entry-label*⟩ has the `category` set to ⟨*category*⟩ (uses `\ifglsfieldeq` for the test).

\glsifcategoryattribute{⟨*category*⟩}{⟨*attribute*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩} §10.2.2; 537

Tests if the given category has the given attribute set to ⟨*value*⟩. Does ⟨*true*⟩ if the attribute is ⟨*value*⟩ and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category.

\glsifcategoryattributehasitem{⟨*category*⟩}{⟨*attribute*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §10.2.2; 538

Does ⟨*true*⟩ if the category has the attribute (whose value is a comma-separated list) contains the given item and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category. The item and list are expanded and passed to datatool's `\DTLifinlist` to perform the test.

\glsifcategoryattributetrue{⟨*category*⟩}{⟨*attribute*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §10.2.2; 538

Tests if the given category has the given attribute set to `true`. Does ⟨*true*⟩ if the attribute is `true` and ⟨*false*⟩ otherwise. Does ⟨*false*⟩ if there's no such attribute for the given category.

\glsifindexed{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §5.8; 273

Tests if the value obtained from `\glsentryindexcount` is greater than 0.

\glsifnotregular{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.04+ §10.2.2; 538

Does ⟨*true*⟩ if the category for the given entry has the `regular` attribute explicitly set to `false`, otherwise does ⟨*false*⟩.

\glsifnotregularcategory{⟨*category*⟩}{⟨*true*⟩}{⟨*false*⟩}
glossaries−extra v1.04+

§10.2.2; 537

Does ⟨*true*⟩ if the given category has the `regular` attribute explicitly set to `false`, otherwise does ⟨*false*⟩.

\glsifplural{⟨*true*⟩}{⟨*false*⟩} glossaries

Initialised by the `\gls`-like and `\glstext`-like commands, this expands to ⟨*true*⟩ if the calling command accesses a plural field (such as `\glspl` or `\glsplural`) otherwise it expands to ⟨*false*⟩. This command may be used within associated hooks, entry display styles (`\defglsentryfmt`), and the post-link hook.

\glsifregular{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}

§10.2.2; 538

Does ⟨*true*⟩ if the category for the given entry has the `regular` attribute explicitly set to `true`, otherwise does ⟨*false*⟩.

\glsifregularcategory{⟨*category*⟩}{⟨*true*⟩}{⟨*false*⟩}

§10.2.2; 537

Does ⟨*true*⟩ if the given category has the `regular` attribute explicitly set to `true`, otherwise does ⟨*false*⟩.

\glsignore{⟨*text*⟩} glossaries v4.12+

Does nothing. When used as a location encap, this signifies to `bib2gls` that the entry is required but the location shouldn't be added to the location list. With other indexing methods, this simply creates an invisible location.

\glsindexingsetting glossaries v4.50+ & glossaries−extra v1.49+

Indicates what indexing option has been chosen.

\glsindexsubgroupitem{ ⟨*previous group level*⟩ } { ⟨*level*⟩ } { ⟨*parent label*⟩ } { ⟨*group label*⟩ } { ⟨*group title*⟩ } glossaries−extra−stylemods v1.49+

Used to format sub-group headers for the indexgroup styles.

\glsinitreunsets glossaries−extra v1.49+ §5.1.1; 193

Hook that initialises the `prereset`, `preunset` and `postunset` settings.

\glsinlinedescformat{ ⟨*description*⟩ } { ⟨*symbol*⟩ } { ⟨*location list*⟩ }
glossary−inline v3.03+

Formats the description, symbol and location list for top-level entries.

\glsinlinesubdescformat{ ⟨*description*⟩ } { ⟨*symbol*⟩ } { ⟨*location list*⟩ }
glossary−inline v3.03+

Formats the description, symbol and location list for child entries.

\glsinnerfmtabbrvfont{ ⟨*text*⟩ } glossaries−extra v1.49+ §4.5.3.1; 173

Robust command that applies both `\glsabbrvfont` and `\glsxtrgenentrytextfmt` to ⟨*text*⟩.

\glsinnerfmtlongfont{ ⟨*text*⟩ } glossaries−extra v1.49+ §4.5.3.1; 173

Applies both `\glslongfont` and `\glsxtrgenentrytextfmt` to ⟨*text*⟩.

\glsinsert glossaries

The final ⟨*insert*⟩ argument passed to the `\gls`-like commands (but not to the `\glstext`-like commands, where the ⟨*insert*⟩ is added to `\glscustomtext`). This command may be used within associated hooks, entry display styles (`\defglsentryfmt`), and the post-link hook.

\glskeylisttok glossaries

§4.5.3.1; 169

A token register that stores the options passed to \newabbreviation.

\glslabel glossaries

The current entry label, initialised by the \gls-like and \glstext-like commands. This command may be used within associated hooks, entry display styles (\defglsentryfmt), and the post-link hook.

\glslabeltok glossaries

§4.5.3.1; 169

A token register that stores the entry's label.

\glsletentryfield{⟨*cs*⟩}{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries v4.07+

Fetches the value of the given field (identified by its internal label ⟨*field-label*⟩) for the entry given by ⟨*entry-label*⟩ and stores it in the command ⟨*cs*⟩.

\Glslink[⟨*options*⟩]{⟨*entry-label*⟩}{⟨*text*⟩} *modifiers:* * + ⟨*alt-mod*⟩
glossaries v4.50+

As \glslink but sets the link text to \glssentencecase{⟨*text*⟩}. This is provided to allow a sentence case mapping in the event that \glslink occurs at the start of content that has automated case-changing.

\glslink[⟨*options*⟩]{⟨*entry-label*⟩}{⟨*text*⟩} *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩ with the given ⟨*text*⟩ as the link text. This command does not alter or depend on the first use flag (use \glsdisp instead, if the first use flag needs to be unset). This command is considered a \glstext-like command. For the first optional argument, see \glslink options.

\glslinkcheckfirsthyperhook glossaries v4.08+

Hook used at the end of the code in the \gls-like commands that tests if the hyperlink should be switched off on first use.

\glslinkpostsetkeys glossaries v4.16+

Hook implemented after setting the options passed to the \gls-like and \glstext-like commands.

\glslinkpresetkeys glossaries−extra v1.26+ §5.1.1; 192

Hook implemented before setting the options passed to the \gls-like and \glstext-like commands.

\glslinkwrcontent{⟨*code*⟩} glossaries−extra v1.48+ §5; 188

Encapsulates the link text and indexing. Just does ⟨*code*⟩ by default.

\glslistchildpostlocation *initial:* **.** glossaries−extra−stylemods v1.21+ §8.6.5.3; 446

Used after the child entry location list for the list styles.

\glslistchildprelocation *initial:* \glslistprelocation
glossaries−extra−stylemods v1.21+ §8.6.5.3; 445

Used before the child entry location list for the list styles.

\glslistdesc{⟨*entry-label*⟩} glossaries−extra−stylemods v1.31+ §8.6.5.3; 445

Used to display the description for the list styles.

\glslistdottedwidth glossary−list

A length register used by listdotted.

\glslistexpandedname{ ⟨*entry-label*⟩ } glossary−list v4.48+

Used by \glslistinit to provide better integration with gettitlestring.

\glslistgroupafterheader glossaries−extra−stylemods v1.47+ §8.6.5.3; 446

Used after group headings in the listgroup styles.

\glslistgroupheaderitem{ ⟨*group-label*⟩ } { ⟨*header code*⟩ } §8.6.5.3; 446
glossaries−extra−stylemods v1.47+

Used to display the group headings in the listgroup styles.

\glslistgroupskip glossaries−extra−stylemods v1.31+ §8.6.5.3; 446

Used for the group skip in the list styles.

\glslistinit glossary−list v4.48+

Used to disable problematic commands at the start the list styles to provide better integration with gettitlestring.

\glslistitem{ ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.47+ §8.6.5.3; 445

Used to display the top-level entry item in the list styles.

\glslistprelocation *initial:* \glsxtrprelocation §8.6.5.3; 445
glossaries−extra−stylemods v1.21+

Used before the top-level entry location list for the list styles.

\glslocalreset{ ⟨*entry-label*⟩ } glossaries

Locally resets the entry's first use flag. That is, this marks the entry as "not used".

\glslocalresetall [⟨*glossary labels list*⟩] glossaries

Locally resets the first use flag for all entries in whose labels are listed in the ⟨*glossary labels list*⟩ comma-separated list. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\glslocalreseteach { ⟨*entry-labels*⟩ } glossaries−extra v1.31+ §5.10; 290

Locally resets each listed entry's first use flag.

\glslocalunset { ⟨*entry-label*⟩ } glossaries

Locally unsets the entry's first use flag. That is, this marks the entry as "used".

\glslocalunsetall [⟨*glossary labels list*⟩] glossaries

Locally unsets the first use flag for all entries in whose labels are listed in the ⟨*glossary labels list*⟩ comma-separated list. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\glslocalunseteach { ⟨*entry-labels*⟩ } glossaries−extra v1.31+ §5.10; 289

Locally unsets each listed entry's first use flag.

\glslongaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp

Does ⟨*text*⟩ with the `longaccess` replacement text (if set).

\glslongdefaultfont { ⟨*text*⟩ } glossaries−extra v1.04+ 139

Formatting command for the long form used by the abbreviation styles that don't apply a font change by default.

\glslongemfont { ⟨*text*⟩ } glossaries−extra v1.04+ 164

Long form font used by the “em” abbreviation styles.

\glslongextraCustomIAlign *initial:* `l` glossary−longextra v1.50+ §8.7.2.7; 478

Expands to the column alignment for the first custom field.

\glslongextraCustomIField *initial:* `useri` glossary−longextra v1.50+ §8.7.2.7; 476

Expands to the internal field name of the first custom field.

\glslongextraCustomIFmt { ⟨*entry-label*⟩ } glossary−longextra v1.50+ §8.7.2.7; 477

The format of the first custom entry.

\glslongextraCustomIHeader glossary−longextra v1.50+ §8.7.2.7; 477

Expands to the header name of the first custom column.

\glslongextraCustomIIAlign *initial:* `l` glossary−longextra v1.50+ §8.7.2.7; 478

Expands to the column alignment for the second custom field.

\glslongextraCustomIIField *initial:* `userii` glossary−longextra v1.50+ §8.7.2.7; 477

Expands to the internal field name of the second custom field.

\glslongextraCustomIIFmt { ⟨*entry-label*⟩ } glossary−longextra v1.50+ §8.7.2.7; 478

The format of the second custom entry.

\glslongextraCustomIIHeader glossary−longextra v1.50+ §8.7.2.7; 477

Expands to the header name of the second custom column.

\glslongextraCustomIIIAlign *initial:* `l` glossary−longextra v1.50+ §8.7.2.7; 479

Expands to the column alignment for the third custom field.

\glslongextraCustomIIIField *initial:* `useriii`
glossary−longextra v1.50+ §8.7.2.7; 477

Expands to the internal field name of the third custom field.

\glslongextraCustomIIIFmt{⟨*entry-label*⟩} glossary−longextra v1.50+ §8.7.2.7; 478

The format of the third custom entry.

\glslongextraCustomIIIHeader glossary−longextra v1.50+ §8.7.2.7; 477

Expands to the header name of the third custom column.

\glslongextraCustomIIINameHeader glossary−longextra v1.50+ §8.7.2.7; 480

The header for the longtable long−custom3−name style.

\glslongextraCustomIIINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 480

The header for the long−custom3−name style.

\glslongextraCustomIIISetDescWidth glossary−longextra v1.50+ §8.7.2.7; 481

Used to set the length `\glsdescwidth` for long−name−custom3−desc style.

\glslongextraCustomIINameHeader glossary−longextra v1.50+ §8.7.2.7; 480

The header for the longtable long−custom2−name style.

\glslongextraCustomIINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the long−custom2−name style.

\glslongextraCustomIISetDescWidth glossary−longextra v1.50+ §8.7.2.7; 481

Used to set the length `\glsdescwidth` for long−name−custom2−desc style.

\glslongextraCustomINameHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the longtable long−custom1−name style.

\glslongextraCustomINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the long−custom1−name style.

\glslongextraCustomISetDescWidth glossary−longextra v1.50+ §8.7.2.7; 481

Used to set the length `\glsdescwidth` for long−name−custom1−desc style.

\glslongextraCustomTabularFooter glossary−longextra v1.50+ §8.7.2.7; 479

The footer for the custom styles.

\glslongextraDescAlign glossary−longextra v1.37+ §8.7.2; 463

The horizontal alignment for the description column.

\glslongextraDescCustomIIINameHeader glossary−longextra v1.50+ §8.7.2.7; 483

The header for the longtable long−desc−custom3−name style.

\glslongextraDescCustomIIINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 483

The header for the long−desc−custom3−name style.

\glslongextraDescCustomIINameHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the longtable long−desc−custom2−name style.

\glslongextraDescCustomIINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the long−desc−custom2−name style.

\glslongextraDescCustomINameHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the longtable long−desc−custom1−name style.

\glslongextraDescCustomINameTabularHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the long−desc−custom1−name style.

\glslongextraDescFmt { ⟨*entry-label*⟩ } glossary−longextra v1.37+ §8.7.2; 463

Used by the glossary−longextra styles to display a top-level entry's description and post-description hook.

\glslongextraDescNameHeader glossary−longextra v1.37+ §8.7.2.1; 466

Sets the header and footer for the long−desc−name style with longtable.

\glslongextraDescNameTabularFooter glossary−longextra v1.37+ §8.7.2.1; 466

Displays the footer for the long−desc−name style.

\glslongextraDescNameTabularHeader glossary−longextra v1.37+ §8.7.2.1; 466

Displays the header for the long−desc−name style.

\glslongextraDescSymHeader glossary−longextra v1.49+ §8.7.2.5; 473

Sets the header and footer for the long−desc−sym style with longtable.

\glslongextraDescSymNameHeader glossary−longextra v1.37+ §8.7.2.2; 468

Sets the header and footer for the long−desc−sym−name style with longtable.

\glslongextraDescSymNameTabularFooter glossary−longextra v1.37+ §8.7.2.2; 468

Displays the footer for the long−desc−sym−name style.

\glslongextraDescSymNameTabularHeader glossary−longextra v1.37+ §8.7.2.2; 468

Displays the header for the long−desc−sym−name style.

\glslongextraDescSymTabularFooter glossary−longextra v1.49+ §8.7.2.5; 473

Displays the footer for the long−desc−sym style.

\glslongextraDescSymTabularHeader glossary−longextra v1.49+ §8.7.2.5; 473

Displays the header for the long−desc−sym style.

\glslongextraGroupHeading{ ⟨*number columns*⟩ } { ⟨*group-label*⟩ }
glossary−longextra v1.37+

§8.7.2; 464

Formats the top-level group heading.

\glslongextraHeaderFmt{ ⟨*text*⟩ } glossary−longextra v1.37+

§8.7.2; 462

Used to format the column headers.

\glslongextraLocationAlign glossary−longextra v1.37+

§8.7.2; 464

The horizontal alignment for the location list column.

\glslongextraLocationDescNameHeader glossary−longextra v1.37+

§8.7.2.3; 469

Sets the header and footer for the long−loc−desc−name style with longtable.

\glslongextraLocationDescNameTabularFooter
glossary−longextra v1.37+

§8.7.2.3; 469

Displays the footer for the long−loc−desc−name style.

\glslongextraLocationDescNameTabularHeader
glossary−longextra v1.37+

§8.7.2.3; 469

Displays the header for the long−loc−desc−name style.

\glslongextraLocationDescSymNameHeader glossary−longextra v1.37+

§8.7.2.4; 471

Sets the header and footer for the long−loc−desc−sym−name style with longtable.

\glslongextraLocationDescSymNameTabularFooter
glossary−longextra v1.37+

§8.7.2.4; 471

Displays the footer for the long−loc−desc−sym−name style.

\glslongextraLocationDescSymNameTabularHeader
glossary−longextra v1.37+

§8.7.2.4; 471

Displays the header for the long−loc−desc−sym−name style.

\glslongextraLocationFmt { ⟨*entry-label*⟩ } { ⟨*location list*⟩ }
glossary−longextra v1.37+

§8.7.2; 463

Used by the glossary−longextra styles to display a top-level entry's location list.

\glslongextraLocationSymDescNameHeader glossary−longextra v1.37+

§8.7.2.4; 470

Sets the header and footer for the long−loc−sym−desc−name style with longtable.

\glslongextraLocationSymDescNameTabularFooter
glossary−longextra v1.37+

§8.7.2.4; 470

Displays the footer for the long−loc−sym−desc−name style.

\glslongextraLocationSymDescNameTabularHeader
glossary−longextra v1.37+

§8.7.2.4; 470

Displays the header for the long−loc−sym−desc−name style.

\glslongextraLocSetDescWidth glossary−longextra v1.37+

§8.7.2.3; 468

Computes the value of `\glsdescwidth` according to the widest name for styles that only show the name, location list and description.

\glslongextraLongFmt { ⟨*entry-label*⟩ } glossary−longextra v1.49+

§8.7.2.6; 474

The formatting for the long form in the abbr−long−short and abbr−short−long styles.

\glslongextraLongHeader *initial:* \descriptionname
glossary−longextra v1.49+ §8.7.2.6; 474

The long column header for the abbr−long−short and abbr−short−long styles.

\glslongextraLongShortHeader glossary−longextra v1.49+ §8.7.2.6; 476

Sets the header and footer for the abbr−short−long style with longtable.

\glslongextraLongShortTabularFooter glossary−longextra v1.49+ §8.7.2.6; 476

Displays the footer for the abbr−short−long style.

\glslongextraLongShortTabularHeader glossary−longextra v1.49+ §8.7.2.6; 476

Displays the header for the abbr−short−long style.

\glslongextraNameAlign *initial:* l glossary−longextra v1.37+ §8.7.2; 462

The horizontal alignment for the name column.

\glslongextraNameCustomIDescHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the longtable long−name−custom1−desc style.

\glslongextraNameCustomIDescTabularHeader
glossary−longextra v1.50+ §8.7.2.7; 482

The header for the long−name−custom1−desc style.

\glslongextraNameCustomIHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the longtable long−name−custom1 style.

\glslongextraNameCustomIIDescHeader glossary−longextra v1.50+ §8.7.2.7; 482

The header for the longtable long−name−custom2−desc style.

\glslongextraNameCustomIIDescTabularHeader
glossary−longextra v1.50+ §8.7.2.7; 482

The header for the long−name−custom2−desc style.

\glslongextraNameCustomIIHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the longtable long−name−custom2 style.

\glslongextraNameCustomIIIDescHeader glossary−longextra v1.50+ §8.7.2.7; 483

The header for the longtable long−name−custom3−desc style.

\glslongextraNameCustomIIIDescTabularHeader
glossary−longextra v1.50+ §8.7.2.7; 483

The header for the long−name−custom3−desc style.

\glslongextraNameCustomIIIHeader glossary−longextra v1.50+ §8.7.2.7; 480

The header for the longtable long−name−custom3 style.

\glslongextraNameCustomIIITabularHeader glossary−longextra v1.50+ §8.7.2.7; 480

The header for the long−name−custom3 style.

\glslongextraNameCustomIITabularHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the long−name−custom2 style.

\glslongextraNameCustomITabularHeader glossary−longextra v1.50+ §8.7.2.7; 479

The header for the long−name−custom1 style.

\glslongextraNameDescHeader glossary−longextra v1.37+ §8.7.2.1; 465

Sets the header and footer for the long−name−desc style with longtable.

\glslongextraNameDescLocationHeader glossary−longextra v1.37+ §8.7.2.3; 468

Sets the header and footer for the long−name−desc−loc style with longtable.

\glslongextraNameDescLocationTabularFooter
glossary−longextra v1.37+ §8.7.2.3; 468

Displays the footer for the long−name−desc−loc style.

\glslongextraNameDescLocationTabularHeader
glossary−longextra v1.37+ §8.7.2.3; 468

Displays the header for the long−name−desc−loc style.

\glslongextraNameDescSymHeader glossary−longextra v1.37+ §8.7.2.2; 466

Sets the header and footer for the long−name−desc−sym style with longtable.

\glslongextraNameDescSymLocationHeader glossary−longextra v1.37+ §8.7.2.4; 470

Sets the header and footer for the long−name−desc−sym−loc style with longtable.

\glslongextraNameDescSymLocationTabularFooter
glossary−longextra v1.37+ §8.7.2.4; 470

Displays the footer for the long−name−desc−sym−loc style.

\glslongextraNameDescSymLocationTabularHeader
glossary−longextra v1.37+

§8.7.2.4; 469

Displays the header for the long−name−desc−sym−loc style.

\glslongextraNameDescSymTabularFooter glossary−longextra v1.37+

§8.7.2.2; 466

Displays the footer for the long−name−desc−sym style.

\glslongextraNameDescSymTabularHeader glossary−longextra v1.37+

§8.7.2.2; 466

Displays the header for the long−name−desc−sym style.

\glslongextraNameDescTabularFooter glossary−longextra v1.37+

§8.7.2.1; 465

Displays the footer for the long−name−desc style.

\glslongextraNameDescTabularHeader glossary−longextra v1.37+

§8.7.2.1; 465

Displays the header for the long−name−desc style.

\glslongextraNameFmt { ⟨*entry-label*⟩ } glossary−longextra v1.37+

§8.7.2; 462

Used by the glossary−longextra styles to add the hypertarget (if supported) and display a top-level entry's name.

\glslongextraNameSymDescHeader glossary−longextra v1.37+

§8.7.2.2; 467

Sets the header and footer for the long−name−sym−desc style with longtable.

\glslongextraNameSymDescLocationHeader glossary−longextra v1.37+

§8.7.2.4; 470

Sets the header and footer for the long−name−sym−desc−loc style with longtable.

\glslongextraNameSymDescLocationTabularFooter
glossary−longextra v1.37+

§8.7.2.4; 470

Displays the footer for the long−name−sym−desc−loc style.

\glslongextraNameSymDescLocationTabularHeader
glossary−longextra v1.37+

§8.7.2.4; 470

Displays the header for the long−name−sym−desc−loc style.

\glslongextraNameSymDescTabularFooter glossary−longextra v1.37+

§8.7.2.2; 467

Displays the footer for the long−name−sym−desc style.

\glslongextraNameSymDescTabularHeader glossary−longextra v1.37+

§8.7.2.2; 467

Displays the header for the long−name−sym−desc style.

\glslongextraSetDescWidth glossary−longextra v1.37+

§8.7.2.1; 465

Computes the value of `\glsdescwidth` according to the widest name for styles that only show the name and description.

\glslongextraSetWidest{⟨*widest-name*⟩} glossary−longextra v1.37+

§8.7.2; 463

Identifies ⟨*widest-name*⟩ as the widest top-level name.

\glslongextraShortHeader *initial:* `\entryname`
glossary−longextra v1.49+

§8.7.2.6; 474

The short column header for the abbr−long−short and abbr−short−long styles.

\glslongextraShortLongHeader glossary−longextra v1.49+

§8.7.2.6; 476

Sets the header and footer for the abbr−short−long style with longtable.

\glslongextraShortLongTabularFooter glossary−longextra v1.49+ §8.7.2.6; 475

Displays the footer for the abbr−short−long style.

\glslongextraShortLongTabularHeader glossary−longextra v1.49+ §8.7.2.6; 475

Displays the header for the abbr−short−long style.

\glslongextraShortNoNameSetDescWidth glossary−longextra v1.49+ §8.7.2.6; 475

Sets the value of `\glsdescwidth` for the abbr−long−short and abbr−short−long styles.

\glslongextraShortTargetFmt { ⟨*entry-label*⟩ } glossary−longextra v1.49+ §8.7.2.6; 474

The formatting, including the target, for the short form in the abbr−long−short and abbr−short−long styles.

\glslongextraSubCustomIFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ }
glossary−longextra v1.50+ §8.7.2.7; 477

The format of the first custom sub-entry.

\glslongextraSubCustomIIFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ }
glossary−longextra v1.50+ §8.7.2.7; 478

The format of the second custom sub-entry.

\glslongextraSubCustomIIIFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ }
glossary−longextra v1.50+ §8.7.2.7; 478

The format of the third custom sub-entry.

\glslongextraSubDescFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } glossary−longextra v1.37+ §8.7.2; 463

Used by the glossary−longextra styles to display a child entry's description and post-description hook.

\glslongextraSubGroupHeading { ⟨*number columns*⟩ } { ⟨*prev group level*⟩ } { ⟨*group level*⟩ } { ⟨*parent-entry-label*⟩ } { ⟨*group-label*⟩ } glossary−longextra v1.49+ §8.7.2; 464

Formats the sub-group heading, if supported.

\glslongextraSubLocationFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossary−longextra v1.37+ §8.7.2; 464

Used by the glossary−longextra styles to display a child entry's location list.

\glslongextraSubLongFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } glossary−longextra v1.49+ §8.7.2.6; 475

The formatting for child entry long forms in the abbr−long−short and abbr−short−long styles.

\glslongextraSubNameFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } glossary−longextra v1.37+ §8.7.2; 462

Used by the glossary−longextra styles to add the hypertarget (if supported) for child-entries. The name isn't shown by default.

\glslongextraSubShortTargetFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } glossary−longextra v1.49+ §8.7.2.6; 475

The formatting, including the target, for child entry short forms in the abbr−long−short and abbr−short−long styles.

\glslongextraSubSymbolFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ } glossary−longextra v1.37+ §8.7.2; 464

Used by the glossary−longextra styles to display a child entry's symbol.

\glslongextraSubSymbolOrName { ⟨*level*⟩ } { ⟨*entry-label*⟩ }
glossary−longextra v1.49+

§8.7.2.5; 472

Adds the hypertarget (if supported) and displays the symbol if set or the name otherwise for child entries.

\glslongextraSubSymbolTargetFmt { ⟨*level*⟩ } { ⟨*entry-label*⟩ }
glossary−longextra v1.49+

§8.7.2.5; 471

Adds the hypertarget (if supported) and displays the symbol for child entries.

\glslongextraSymbolAlign *initial:* c glossary−longextra v1.37+

§8.7.2; 464

The horizontal alignment for the symbol column.

\glslongextraSymbolFmt { ⟨*entry-label*⟩ } glossary−longextra v1.37+

§8.7.2; 464

Used by the glossary−longextra styles to display a top-level entry's symbol.

\glslongextraSymbolNameAlign *initial:* l glossary−longextra v1.49+

§8.7.2.5; 471

The horizontal alignment for the symbol column when it's being used instead of the name.

\glslongextraSymbolOrName { ⟨*entry-label*⟩ } glossary−longextra v1.49+

§8.7.2.5; 472

Adds the hypertarget (if supported) and displays the symbol if set or the name otherwise for top-level entries.

\glslongextraSymbolTargetFmt { ⟨*entry-label*⟩ } glossary−longextra v1.49+

§8.7.2.5; 471

Adds the hypertarget (if supported) and displays the symbol for top-level entries.

\glslongextraSymDescHeader glossary−longextra v1.49+

§8.7.2.5; 473

Sets the header and footer for the long−sym−desc style with longtable.

\glslongextraSymDescNameHeader glossary−longextra v1.37+ §8.7.2.2; 467

Sets the header and footer for the long−sym−desc−name style with longtable.

\glslongextraSymDescNameTabularFooter glossary−longextra v1.37+ §8.7.2.2; 467

Displays the footer for the long−sym−desc−name style.

\glslongextraSymDescNameTabularHeader glossary−longextra v1.37+ §8.7.2.2; 467

Displays the header for the long−sym−desc−name style.

\glslongextraSymDescTabularFooter glossary−longextra v1.49+ §8.7.2.5; 473

Displays the footer for the long−sym−desc style.

\glslongextraSymDescTabularHeader glossary−longextra v1.49+ §8.7.2.5; 472

Displays the header for the long−sym−desc style.

\glslongextraSymLocSetDescWidth glossary−longextra v1.37+ §8.7.2.4; 469

Computes the value of `\glsdescwidth` according to the widest name for styles that show the name, symbol, location list and description.

\glslongextraSymNoNameSetDescWidth glossary−longextra v1.49+ §8.7.2.5; 472

Computes the value of `\glsdescwidth` according to the widest name for styles that only show the symbol and description.

\glslongextraSymSetDescWidth glossary−longextra v1.37+ §8.7.2.2; 466

Computes the value of `\glsdescwidth` according to the widest name for styles that only show the name, symbol and description.

\glslongextraTabularVAlign *initial:* c glossary−longextra v1.37+ §8.7.2; 462

Only for use with the tabular setting, this should expand to the tabular environment's vertical alignment specifier.

\glslongextraUpdateWidest { ⟨*name*⟩ } glossary−longextra v1.37+ §8.7.2; 463

If ⟨*name*⟩ is wider than the current widest name, it will be set as the new widest name.

\glslongextraUpdateWidestChild { ⟨*level*⟩ } { ⟨*name*⟩ }
glossary−longextra v1.37+ §8.7.2; 463

As \glslongextraUpdateWidest but for child entries. Does nothing by default.

\GlsLongExtraUseTabularfalse glossary−longextra v1.37+ §8.7.2; 461

Sets \ifGlsLongExtraUseTabular to false (if this setting is required, the style must be set after this command).

\GlsLongExtraUseTabulartrue glossary−longextra v1.37+ §8.7.2; 461

Sets \ifGlsLongExtraUseTabular to true (if this setting is required, the style must be set after this command).

\glslongfont { ⟨*text*⟩ } 177

Font formatting command for the long form, initialised by the abbreviation style.

\glslongfootnotefont { ⟨*text*⟩ } glossaries−extra v1.05+ 151

Formatting command for the long form used by the footnote abbreviation styles.

\glslonghyphenfont { ⟨*text*⟩ } glossaries−extra v1.17+ 154

Long form font used by the "hyphen" abbreviation styles.

\glslongonlyfont{⟨*text*⟩} glossaries−extra v1.17+ 160

Long form font used by the "only" abbreviation styles.

\glslongpltok §4.5.3.1; 170

A token register that stores the long plural form (which may have been modified after being passed to `\newabbreviation`).

\glslongpluralaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−accsupp

Does ⟨*text*⟩ with the `longpluralaccess` replacement text (if set).

\glslongtok glossaries §4.5.3.1; 169

A token register that stores the long form (which may have been modified after being passed to `\newabbreviation`).

\glslonguserfont{⟨*text*⟩} glossaries−extra v1.04+ 142

Long form font used by the "user" abbreviation styles.

\glslowercase{⟨*text*⟩} glossaries v4.50+ §5.2.2; 204

Converts ⟨*text*⟩ to lowercase.

\glsmakefirstuc{⟨*text*⟩} mfirstuc v1.05+

Used by `\makefirstuc` to perform the actual case-change. As from mfirstuc v2.08+ this just uses `\MFUsentencecase`.

\glsmfuaddmap{⟨*cs1*⟩}{⟨*cs2*⟩} glossaries v4.50+ & glossaries−extra v1.49+ §5.2.1; 204

If mfirstuc v2.08+ is installed, this will use `\MFUaddmap`, otherwise it will use `\glsmfuexcl` instead. See §5.2.1 for further details.

\glsmfublocker{⟨*cs*⟩} glossaries v4.50+ & glossaries−extra v1.49+ §5.2.1; 203

If mfirstuc v2.08+ is installed, this will use `\MFUblocker`, otherwise it will use `\glsmfuexcl` instead. See §5.2.1 for further details.

\glsmfuexcl{⟨*cs*⟩} glossaries v4.50+ & glossaries−extra v1.49+ §5.2.1; 202

If mfirstuc v2.08+ is installed, this will use `\MFUexcl`, otherwise it will implement something similar. See §5.2.1 for further details.

\GLSname[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As `\glsname` but converts the link text to all caps. This command is incompatible with some abbreviation styles.

\Glsname[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As `\glsname` but converts the link text to sentence case. Use `\Glossentryname` within custom glossary styles instead of this command.

\glsname[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `name` value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options. Use `\glossentryname` within custom glossary styles instead of this command.

\glsnameaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−accsupp

Does ⟨*text*⟩ with the `access` replacement text (if set).

\glsnamefont { ⟨*text*⟩ } glossaries

Used by \glossentryname to apply a font change to the name, unless (with glossaries −extra) the glossnamefont attribute has been set.

\glsnextpages glossaries v3.0+

Designed for use with makeindex and xindy, this may be placed in an entry's description to override nonumberlist.

\glsnoexpandfields glossaries v3.08a+

Don't expand field values when defining entries, except for those that explicitly have expansion enabled with \glssetexpandfield.

\glsnoidxdisplayloc { ⟨*prefix*⟩ } { ⟨*counter*⟩ } { ⟨*format*⟩ } { ⟨*location*⟩ }
glossaries v4.04+

§11.6.6; 605

Used to display a location in the location list.

\glsnonextpages glossaries v1.17+

Designed for use with makeindex and xindy, this may be placed in an entry's description to suppress the entry's location list.

\glsnumberformat { ⟨*location(s)*⟩ } glossaries

The default format for entry locations. If hyperlinks are defined, this will use \glshypernumber otherwise it will simply display its argument, which may be a single location, or locations delimited by \delimR or \delimN.

\glsnumbersgroupname *initial:* Numbers glossaries
(language-sensitive)

Expands to the title of the numbers group and (if the numbers package option is used) the numbers glossary.

\glspagelistwidth glossary−long & glossary−super

A length register used to set the width of the location list column for tabular-like styles.

\glspar glossaries

Paragraph break (for instances where \par can't be used directly).

\glspatchLToutput glossary−longbooktabs v4.21+

Applies a patch to longtable to check for instances of the group skip occurring at a page break.

\glspdffmtfull{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 214

Shortcut for \glsentrylong{⟨*entry-label*⟩} (\glsentryshort{⟨*entry-label*⟩}) for use in PDF bookmarks or other text-only contexts.

\glspdffmtfullpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.2; 214

Shortcut for \glsentrylongpl{⟨*entry-label*⟩} (\glsentryshortpl{⟨*entry-label*⟩}) for use in PDF bookmarks or other text-only contexts.

\glspdfsentencecase{⟨*text*⟩} glossaries−extra v1.54+ §5.2.4; 205

Provided primarily for use in the second argument of \texorpdfstring, this is a shortcut that expands ⟨*text*⟩ before applying \MFUsentencecase.

\glspenaltygroupskip glossary−longbooktabs v4.21+

The definition of \glsgroupskip with nogroupskip=false for the glossary−longbooktabs styles.

\glspercentchar glossaries v4.10+

Expands to a literal percent sign.

\GLSpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glspl but converts the link text to all caps.

\Glspl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glspl but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glspl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \gls but uses the relevant plural form.

\GLSplural [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsplural but converts the link text to all caps. If you have defined the entry with \newabbreviation use \GLSxtrshortpl or \GLSpl[preunset] instead.

\Glsplural [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glsplural but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc. If you have defined the entry with \newabbreviation use \Glsxtrshortpl or \Glspl[preunset] instead.

\glsplural [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the plural value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \newabbreviation use \glsxtrshortpl for the short form or \gls[preunset], as some

abbreviation styles are too complicated to work with \glsplural. For the first optional argument, see \glslink options.

\glspluralaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp

Does ⟨*text*⟩ with the pluralaccess replacement text (if set).

\glspluralsuffix

Expands to the letter "s" and is used to form default plurals. This command isn't language-sensitive as there's guarantee when it will be expanded. (It may be expanded when the entry is defined or it may be expanded when the entry is used). If you need to suppress this suffix for abbreviations, use the noshortplural attribute. If you need an apostrophe before the "s" for single-letter abbreviations to avoid ambiguity, use the aposplural attribute.

\glspostdescription glossaries

A hook that is usually placed after the description in glossary styles. Some of the styles provided with the glossaries package don't use this hook. The glossaries−extra−stylemods redefines those styles to include the hook. The default definition of this command tests for the nopostdot option, but the postpunc option redefines the command to implement the chosen punctuation.

\glspostinline glossary−inline v3.03+

Used at the end of the theglossary environment.

\glspostinlinedescformat { ⟨*description*⟩ } { ⟨*symbol*⟩ } { ⟨*location list*⟩ }
glossary−inline v3.03+

Formats the top-level entry's description, symbol and location list.

\glspostinlinesubdescformat { ⟨*description*⟩ } { ⟨*symbol*⟩ } { ⟨*location list*⟩ } glossary−inline v3.03+

Formats the child entry's description, symbol and location list.

\glspostlinkhook glossaries v4.16+

A post-link hook used after all the \gls-like and \glstext-like commands. This is redefined by glossaries−extra to use \glsxtrpostlinkhook.

\glsprefixsep *initial: empty* glossaries−prefix v4.45+

Separator between the prefix and the term.

\glsprestandardsort { ⟨*sort cs*⟩ } { ⟨*type*⟩ } { ⟨*entry-label*⟩ } glossaries v3.13a+

Hook used with sort=standard to adjust the default sort value (with \makeglossaries or \makenoidxglossaries only).

\glspretopostlink { ⟨*category*⟩ } { ⟨*code*⟩ } glossaries−extra v1.49+ §5.5.4; 256

Prepends ⟨*code*⟩ to post-link hook associated with the category identified by the label ⟨*category*⟩ (or simply defines it, if it doesn't already exist).

\GLSps { ⟨*entry-label*⟩ } glossaries−extra v1.51+ §5.4; 236

Shortcut for \GLSxtrp{short}{⟨*entry-label*⟩}.

\Glsps { ⟨*entry-label*⟩ } glossaries−extra v1.51+ §5.4; 235

Shortcut for \Glsxtrp{short}{⟨*entry-label*⟩}.

\glsps { ⟨*entry-label*⟩ } glossaries−extra v1.07+ §5.4; 235

Shortcut for \glsxtrp{short}{⟨*entry-label*⟩}.

\GLSpt { ⟨*entry-label*⟩ } glossaries−extra v1.51+ §5.4; 236

Shortcut for \GLSxtrp{text}{⟨*entry-label*⟩}.

\Glspt{⟨*entry-label*⟩} glossaries−extra v1.51+ §5.4; 236

Shortcut for \Glsxtrp{text}{⟨*entry-label*⟩}.

\glspt{⟨*entry-label*⟩} glossaries−extra v1.07+ §5.4; 235

Shortcut for \glsxtrp{text}{⟨*entry-label*⟩}.

\glsrefentry{⟨*entry-label*⟩} glossaries v3.0+

References (using \ref) the entry counter or sub-counter (if entrycounter or subentrycounter options are set) otherwise just does \gls{⟨*entry-label*⟩}.

\glsrenewcommand{⟨*cs*⟩}[⟨*n*⟩][⟨*default*⟩]{⟨*definition*⟩} *modifier:* *
glossaries−extra−bib2gls v1.37+ §11.6.8; 622

Like \renewcommand but only issues a warning instead of an error if the command hasn't been defined.

\glsreset{⟨*entry-label*⟩} glossaries

Globally resets the entry's first use flag. That is, this marks the entry as "not used".

\glsresetall[⟨*types*⟩] glossaries

Globally resets all entries associated with the listed glossaries or all glossaries if ⟨*types*⟩ is omitted.

\glsresetcurrcountfalse glossaries−extra v1.49+ §6.1; 318

Sets \ifglsresetcurrcount to false.

\glsresetcurrcounttrue glossaries−extra v1.49+ §6.1; 318

Sets \ifglsresetcurrcount to true.

\glsresetentrylist glossaries

Inserted into the glossary code to counteract the effect of \glsnonextpages.

\glsSavedGlossaryGroup{⟨*ref-label*⟩}{⟨*glossary-label*⟩}{⟨*level*⟩}
{⟨*parent-entry*⟩}{⟨*group-label*⟩}{⟨*group-title*⟩} glossaries v5.1+

This command is written to the aux file by \glswriteglossarygroup and \glswriteglossarysubgroup if saveglossarygroups=true.

\glssee[⟨*tag*⟩]{⟨*entry-label*⟩}{⟨*xr-list*⟩} glossaries v1.17+

Indexes the entry identified by ⟨*entry-label*⟩ as a general cross-reference to the entries identified in the comma-separated list ⟨*xr-list*⟩. The optional argument is the textual tag that's inserted before the cross-reference list and defaults to \seename.

\glsseefirstitem{⟨*entry-label*⟩} glossaries−extra v1.47+ §5.13; 309

Used by \glsseelist to format the first entry.

\glsseeformat[⟨*tag*⟩]{⟨*xr-list*⟩}{⟨*location*⟩} glossaries v1.17+

Used to format the see cross-reference in the location list. This requires a location argument for makeindex even though it isn't required. The default definition is \emph{⟨*tag*⟩} \glsseelist{⟨*xr-list*⟩}.

\glsseeitem{⟨*entry-label*⟩} glossaries v1.17+

Used by \glsseelist to format each entry.

\glsseeitemformat{⟨*entry-label*⟩} glossaries v3.0+ §5.13; 308

Used by \glsseeitem to produce the hyperlink text.

\glsseelastoxfordsep glossaries−extra v1.47+ §5.13; 309

Used by \glsseelist as a separator between penultimate and final entry in the list if there are at least three entries in the list.

\glsseelastsep glossaries v1.17+ §5.13; 309

Used by \glsseelist as a separator between penultimate and final entry in the list.

\glsseelist{⟨*csv-list*⟩} glossaries v1.17+ §5.13; 307

Iterates over a comma-separated list of entry labels ⟨*csv-list*⟩ and formats them. Each label in the list is encapsulated with \glsseeitem (or \mglsseeitem, the label corresponds to a multi-entry). The separators are \glsseelastsep (between the penultimate and last items) and \glsseesep (between all the other items). With glossaries−extra, the first label is encapsulated with \glsseefirstitem (or \mglsseefirstitem) and the final separator for a list consisting of at least three items is given by \glsseelastoxfordsep.

\glsseesep *initial:* , ␣ glossaries v1.17+ §5.13; 309

Used by \glsseelist as a separator between each entry except the last pair.

\glssentencecase{⟨*text*⟩} glossaries v4.50+ & glossaries−extra v1.49+ §5.2.1; 201

Used by sentence case commands, such as \Gls, to perform the case change. This is simply defined to use \makefirstuc.

\glssetabbrvfmt{⟨*category*⟩} §4.5.2; 164

Implements the ⟨*display definitions*⟩ code for the abbreviation style associated with the given category.

\glssetattribute{⟨*entry-label*⟩}{⟨*attribute*⟩}{⟨*value*⟩} §10.2.2; 536

Locally sets the given attribute to ⟨*value*⟩ for the category associated with the entry identified by ⟨*entry-label*⟩.

\glssetcategoriesattribute{ ⟨*category list*⟩ } { ⟨*attribute*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.48+

§10.2.2; 535

Globally sets the given attribute to ⟨*value*⟩ for all the categories in the comma-separated list ⟨*category list*⟩.

\glssetcategoriesattributes{ ⟨*category list*⟩ } { ⟨*attribute list*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.48+

§10.2.2; 536

Globally sets each attribute in the comma separated ⟨*attribute list*⟩ to ⟨*value*⟩ for each category in the comma-separated list ⟨*category list*⟩.

\glssetcategoryattribute{ ⟨*category*⟩ } { ⟨*attribute*⟩ } { ⟨*value*⟩ }

§10.2.2; 535

Locally sets the given attribute to ⟨*value*⟩ for the given category.

\glssetcategoryattributes{ ⟨*category*⟩ } { ⟨*attribute list*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.49+

§10.2.2; 536

Globally sets each attribute in the comma separated ⟨*attribute list*⟩ to ⟨*value*⟩ for the given ⟨*category*⟩.

\glssetcombinedsepabbrvnbsp glossaries−extra v1.48+

§7.4; 355

Defines the multi-entry separators to use a non-breaking space (~) for abbreviations.

\glssetcombinedsepabbrvnone glossaries−extra v1.48+

§7.4; 356

Defines the multi-entry separators to use no separator for abbreviations.

\glssetcombinedsepnarrow{ ⟨*width*⟩ } { ⟨*narrow-sep*⟩ } glossaries−extra v1.48+

§7.4; 357

Defines the multi-entry separators to use ⟨*narrow-sep*⟩ if the width of associated field values is less than ⟨*width*⟩.

\glssetexpandfield{ ⟨*field*⟩ } glossaries v3.13a+

Expand the value of the field identified by its internal field label when defining entries (overrides `\glsnoexpandfields`).

\glssetnoexpandfield{ ⟨*field*⟩ } glossaries v3.13a+

Don't expand the value of the field identified by its internal field label when defining entries (overrides `\glsexpandfields`).

\glssetregularcategory{ ⟨*category*⟩ }

§10.2.2; 536

Locally sets the `regular` attribute to `true` for the given category.

\glssetwidest [⟨*level*⟩] { ⟨*name*⟩ } glossary−tree

Indicates that ⟨*name*⟩ is the widest name for the given hierarchical level.

\glsshortaccessdisplay{ ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp

Does ⟨*text*⟩ with the `shortaccess` replacement text (if set).

\glsshortaccsupp{ ⟨*replacement*⟩ } { ⟨*content*⟩ } glossaries−accsupp v4.45+

Applies ⟨*replacement*⟩ as the expansion (E) attribute for ⟨*content*⟩ using `\glsaccessibility`.

\glsshortpltok

§4.5.3.1; 169

A token register that stores the short plural form (which may have been modified after being passed to `\newabbreviation`).

\glsshortpluralaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩}
glossaries−accsupp

Does ⟨*text*⟩ with the `shortpluralaccess` replacement text (if set).

\glsshorttok glossaries

§4.5.3.1; 169

A token register that stores the short form (which may have been modified after being passed to `\newabbreviation`).

\glsshowtarget{⟨*target-name*⟩} glossaries v4.32+

Formats the target name when `debug=showtargets` is enabled using either `\glsshowtargetinner` or `\glsshowtargetouter`, depending on the current mode.

\glsshowtargetfont glossaries v4.45+

Font declaration used by debugging annotations.

\glsshowtargetfonttext{⟨*text*⟩} glossaries v4.50+

Text-block command that checks for math mode and switches to the font given by the `\glsshowtargetfont` declaration.

\glsshowtargetinner{⟨*target-name*⟩} glossaries v4.50+

Formats the target name for inner and maths mode when `debug=showtargets` is enabled.

\glsshowtargetinnersymleft{name} glossaries−extra v1.48+

§2.5; 31

Shows the left inner annotation followed by the left marker symbol `\glsxtrshowtargetsymbolleft`.

\glsshowtargetinnersymright{name} glossaries−extra v1.48+ §2.5; 31

Shows the right marker symbol \glsxtrshowtargetsymbolright followed by the right inner annotation.

\glsshowtargetouter{⟨*target-name*⟩} glossaries v4.50+

Formats the target name for outer mode when debug=showtargets is enabled. This places a marker (\glsshowtargetsymbol) in the text and ⟨*target-name*⟩ in the margin.

\glsshowtargetsymbol glossaries v4.45+

Marker (▹) used in debugging annotations.

\glsstartrange[⟨*options*⟩]{⟨*entry label list*⟩} glossaries−extra v1.50+ §5.8; 267

Essentially does \glsaddeach[⟨*options*⟩,format=(⟨*encap*⟩]{⟨*entry label list*⟩} where ⟨*encap*⟩ can either be provided by the format key in ⟨*options*⟩ or will default to the format given in \GlsXtrSetDefaultRangeFormat.

\glssubentryitem{⟨*entry-label*⟩} glossaries v3.0+

Does nothing if subentrycounter=false, otherwise increments and displays the associated counter.

\glssubgroupheading{⟨*previous level*⟩}{⟨*level*⟩}{⟨*parent−label*⟩}{⟨*group−label*⟩} glossaries−extra v1.49+ §8.4.1; 405

Used to format sub-group headings. Only applicable with the "unsrt" family of commands. This command won't occur in glossaries that use \printglossary or \printnoidxglossary.

\GLSsymbol[⟨*options*⟩]{⟨*entry-label*⟩}[⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glssymbol but converts the link text to all caps.

\Glssymbol [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As \glssymbol but converts the link text to sentence case. Use \Glossentrysymbol within custom glossary styles instead of this command.

\glssymbol [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the symbol value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options. Use \glossentrysymbol within custom glossary styles instead of this command.

\glssymbolaccessdisplay {⟨*text*⟩} {⟨*entry-label*⟩} glossaries−accsupp

Does ⟨*text*⟩ with the symbolaccess replacement text (if set).

\glssymbolplural [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries

As \glssymbol but for the symbolplural field.

\glssymbolpluralaccessdisplay {⟨*text*⟩} {⟨*entry-label*⟩}
glossaries−accsupp

Does ⟨*text*⟩ with the symbolpluralaccess replacement text (if set).

\glssymbolsgroupname *initial:* Symbols glossaries
(language-sensitive)

Expands to the title of the symbols group and (if the symbols package option is used) the symbols glossary.

\glstableblocksubentry{⟨*entry-label*⟩} glossary−table v1.49+

Displays the child entry identified by ⟨*entry-label*⟩. This command is redefined by block styles.

\glstableblocksubentrysep glossary−table v1.49+ §8.7.4.1; 497

Separator used by `\glstableChildEntries` between child entries.

\glstableblockwidth glossary−table v1.49+ §8.7.4.4; 505

Length register used for the width of each block with `par=justified` or `par=ragged`. Set by the block style.

\glstablecaption{⟨*lot title*⟩}{⟨*title*⟩}{⟨*label code*⟩} glossary−table v1.49+ §8.7.4.2; 498

Produces the caption for the first page of the table.

\glstablecenteralign{⟨*width*⟩} glossary−table v1.49+ §8.7.4.4; 503

Expands to `c` or `p{`⟨*width*⟩`}` or `>\protect\centeringp{`⟨*width*⟩`}`, depending on the `par` setting.

\glstableChildEntries{⟨*entry-label*⟩} glossary−table v1.49+ §8.7.4.1; 496

Iterates over the `childlist` field and formats each child entry in the list for use in the block styles. Does nothing if the list is empty.

\glstabledesccolalign glossary−table v1.49+ §8.7.4.4; 504

Expands to the alignment of the description column.

\glstableDescFmt{⟨*text*⟩} glossary−table v1.50+ §8.7.4.4; 506

Formatting applied to the description.

\glstabledescheader glossary−table v1.49+ §8.7.4.2; 499

Header for the description column.

\glstabledescwidth glossary−table v1.49+ §8.7.4.4; 504

Length register used for the width of the description column with `par=justified` or `par=ragged`. Set by the block style.

\glstableHeaderFmt{⟨*text*⟩} glossary−table v1.49+ §8.7.4.4; 507

Formats the header.

\glstableiffilter{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩} glossary−table v1.49+ §8.7.4; 492

Internally used by the custom handler in `\printunsrttable` to perform additional filtering. This command should do ⟨*true*⟩ if the entry identified by ⟨*entry-label*⟩ should be filtered and ⟨*false*⟩ otherwise.

\glstableiffilterchild{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
glossary−table v1.50+ §8.7.4.1; 496

Internally used by `\glstableChildEntries` to filter child entries. This command should do ⟨*true*⟩ if the child entry identified by ⟨*entry-label*⟩ should be filtered and ⟨*false*⟩ otherwise.

\glstableifhasotherfield{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
glossary−table v1.50+ §8.7.4.4; 507

Expands to ⟨*true*⟩ if the other field is non-void for the given entry otherwise expands to ⟨*false*⟩.

\glstableleftalign{⟨*width*⟩} glossary−table v1.49+ §8.7.4.4; 503

Expands to `l` or `p{`⟨*width*⟩`}` or `>\protect\raggedrightp{`⟨*width*⟩`}`, depending on the `par` setting.

\glstablenamecolalign glossary−table v1.49+ §8.7.4.4; 504

Expands to the alignment of the name column.

\glstableNameFmt { ⟨*text*⟩ } glossary−table v1.50+ §8.7.4.4; 505

Formatting applied to the name.

\glstablenameheader glossary−table v1.49+ §8.7.4.2; 499

Header for the name column.

\glstablenamewidth glossary−table v1.49+ §8.7.4.4; 504

Length register used for the width of the name column with `par=justified` or `par=ragged`. Set by the block style.

\glstablenewline glossary−table v1.50+ §8.7.4.4; 503

Used to start a new row.

\glstablenextcaption { ⟨*lot title*⟩ } { ⟨*title*⟩ } glossary−table v1.49+ §8.7.4.2; 498

Produces the caption for following pages of the table.

\glstableOther { ⟨*entry-label*⟩ } glossary−table v1.50+ §8.7.4.4; 506

Used to display the other field.

\glstableothercolalign glossary−table v1.49+ §8.7.4.4; 504

Expands to the alignment of the other column.

\glstableotherfield *initial: empty* glossary−table v1.49+ §8.7.4.4; 506

Expands to the internal field label of the other field.

\glstableOtherFmt{⟨*text*⟩} glossary−table v1.50+ §8.7.4.4; 506

Formatting applied to the other field.

\glstableotherheader glossary−table v1.49+ §8.7.4.2; 499

Header for the other column.

\glstableotherwidth glossary−table v1.50+ §8.7.4.4; 505

Length register used for the width of the other column with `par=justified` or `par=ragged`. Set by the block style.

\glstablepostnextcaption *initial:* ␣`Cont./` glossary−table v1.49+ §8.7.4.2; 498

Appended to the caption in `\glstablenextcaption`.

\glstablepostpreambleskip *initial:* `5pt` glossary−table v1.50+ §8.7.4.4; 505

Length register that specifies the vertical skip after the preamble.

\glstablePreChildren glossary−table v1.49+ §8.7.4.1; 496

Code performed by `\glstableChildEntries` before the child list.

\glstableprepostambleskip *initial:* `5pt` glossary−table v1.50+ §8.7.4.4; 505

Length register that specifies the vertical skip before the postamble.

\glstablerightalign{⟨*width*⟩} glossary−table v1.49+ §8.7.4.4; 503

Expands to `r` or `p{`⟨*width*⟩`}` or `>\protect\raggedleftp{`⟨*width*⟩`}`, depending on the `par` setting.

\glstablesetstyle{⟨*style-name*⟩} glossary−table v1.49+ §8.7.4.3; 500

Sets the default block style.

\glstableSubDescFmt{⟨*text*⟩} glossary−table v1.50+ §8.7.4.4; 506

Formatting applied to the child description.

\glstablesubentryalign glossary−table v1.50+ §8.7.4.1; 497

Expands to the column alignment used by glstablesubentries.

\glstableSubNameFmt{⟨*text*⟩} glossary−table v1.50+ §8.7.4.4; 505

Formatting applied to the child name.

\glstableSubOther{⟨*entry-label*⟩} glossary−table v1.50+ §8.7.4.4; 507

Used to display the sub-entry other field.

\glstableSubOtherFmt{⟨*text*⟩} glossary−table v1.50+

Formatting applied to the other field.

\glstableSubSymbolFmt{⟨*text*⟩} glossary−table v1.50+ §8.7.4.4; 506

Formatting applied to the child symbol.

\glstablesymbolcolalign glossary−table v1.49+ §8.7.4.4; 504

Expands to the alignment of the symbol column.

\glstableSymbolFmt { ⟨*text*⟩ } glossary−table v1.50+ §8.7.4.4; 505

Formatting applied to the symbol.

\glstablesymbolheader glossary−table v1.49+ §8.7.4.2; 499

Header for the symbol column.

\glstablesymbolwidth glossary−table v1.49+ §8.7.4.4; 505

Length register used for the width of the symbol column with `par=justified` or `par=ragged`. Set by the block style.

\glstarget { ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossaries v1.18+

Used by glossary styles to create a hypertarget (if enabled) for the entry (identified by ⟨*entry-label*⟩). The ⟨*text*⟩ is usually `\glossentryname{`⟨*entry-label*⟩`}`, but it can be something else.

\GLStext [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As `\glstext` but converts the link text to all caps. If you have defined the entry with `\newabbreviation` use `\GLSxtrshort` or `\GLS[preunset]` instead.

\Glstext [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

As `\glstext` but converts the first character of the link text to uppercase (for the start of a sentence) using `\makefirstuc`. If you have defined the entry with `\newabbreviation` use `\Glsxtrshort` or `\Gls[preunset]` instead.

\glstext [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `text` value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with `\newabbreviation` use `\glsxtrshort` for the short form or `\gls[preunset]`, as some abbreviation styles are too complicated to work with `\glstext`. For the first optional argument, see `\glslink` options.

\glstextaccessdisplay{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−accsupp

Does ⟨*text*⟩ with the `textaccess` replacement text (if set).

\glstextformat{⟨*text*⟩} glossaries v1.04+

The default outer text formatting command used by the `\gls`-like and `\glstext`-like commands.

\glstextup{⟨*text*⟩} glossaries v3.09a+

Counteracts the effect of `\textsc`.

\glstildechar glossaries v4.10+

Expands to a literal tilde character.

\glstopicAssignSubIndent{⟨*level*⟩} glossary−topic v1.40+ §8.7.3; 490

Used to set the indentation for sub-levels.

\glstopicAssignWidest{⟨*level*⟩} glossary−topic v1.40+ §8.7.3; 490

Used by `\glstopicAssignSubIndent` to calculate the width of the widest name for the given level.

\glstopicCols glossary−topic v1.40+ §8.7.3; 486

Expands to the number of columns for topicmcols.

\glstopicColsEnv glossary−topic v1.40+ §8.7.3; 484

Expands to the multicols environment name to use for topicmcols.

\glstopicDesc{ ⟨*entry-label*⟩ } glossary−topic v1.40+ §8.7.3; 490

Used to format the top-level description.

\glstopicGroupHeading{ ⟨*group-label*⟩ } glossary−topic v1.40+ §8.7.3; 488

Used to format the top-level group headings, if required.

\glstopicInit glossary−topic v1.40+ §8.7.3; 488

Initialisation hook.

\glstopicItem{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossary−topic v1.40+ §8.7.3; 489

Used to format top-level entries.

\glstopicLoc{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossary−topic v1.40+ §8.7.3; 490

Used to format the top-level location list.

\glstopicMarker{ ⟨*entry-label*⟩ } glossary−topic v1.40+ §8.7.3; 489

Hook inserted before a top-level entry.

\glstopicMidSkip glossary−topic v1.40+ §8.7.3; 489

Vertical space inserted before the description for a top-level entry.

\glstopicParIndent glossary−topic v1.40+ §8.7.3; 488

Length register used for the top-level paragraph indent.

\glstopicPostSkip glossary−topic v1.40+ §8.7.3; 490

Vertical space inserted after the description for a top-level entry.

\glstopicPreSkip glossary−topic v1.40+ §8.7.3; 489

Vertical space inserted before a top-level entry.

\glstopicSubGroupHeading{ ⟨*prev group level*⟩ } { ⟨*group level*⟩ } { ⟨*parent entry*⟩ } { ⟨*group-label*⟩ } glossary−topic v1.49+ §8.7.3; 489

Used to format the sub-group headings, if supported.

\glstopicSubIndent glossary−topic v1.40+ §8.7.3; 488

Length register used for the child indent.

\glstopicSubItem{ ⟨*level*⟩ } { ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossary−topic v1.40+ §8.7.3; 490

Used to format child entries.

\glstopicSubItemBox{ ⟨*level*⟩ } { ⟨*text*⟩ } glossary−topic v1.40+ §8.7.3; 490

Horizontal box used for child name if a widest name has been provided.

\glstopicSubItemParIndent glossary−topic v1.46+ §8.7.3; 488

Length register used for the child paragraph indent.

\glstopicSubItemSep glossary−topic v1.40+ §8.7.3; 490

Horizontal separator used after child names.

\glstopicSubLoc{ ⟨*entry-label*⟩ }{ ⟨*location list*⟩ } glossary−topic v1.41+ §8.7.3; 491

Formats the child location lists.

\glstopicSubNameFont{ ⟨*text*⟩ } glossary−topic v1.40+ §8.7.3; 490

Font command to apply to the child entry name.

\glstopicSubPreLocSep glossary−topic v1.41+ §8.7.3; 491

Separator before the child location lists.

\glstopicTitle{ ⟨*entry-label*⟩ } glossary−topic v1.40+ §8.7.3; 489

Used to format the name and (if provided) symbol for the top-level entry title.

\glstopicTitleFont{ ⟨*text*⟩ } glossary−topic v1.40+ §8.7.3; 489

Font command to apply to the top-level entry title.

\glstreechilddesc{ ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.31+ §8.6.5.4; 449

Displays the given child entry's description with pre and post hooks for the tree styles.

\glstreeChildDescLoc{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ }
glossaries−extra−stylemods v1.41+

§8.6.5.4; 449

Formats the child description (if set) and location list for the tree styles.

\glstreechildpredesc glossary−tree v4.26+

§8.6.5.4; 447

Space inserted before child descriptions.

\glstreechildprelocation *initial:* \glstreeprelocation
glossaries−extra−stylemods v1.21+

§8.6.5.4; 448

Used before the child entry location list for the tree and index styles.

\glstreechildsymbol{ ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.31+

§8.6.5.4; 449

Displays the top-level symbol in parentheses, if set, for the tree styles.

\glstreedefaultnamefmt{ ⟨*text*⟩ } glossaries−extra−stylemods v1.31+

§8.6.5.4; 447

Used as the default name format for the tree and index styles.

\glstreedesc{ ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.31+

§8.6.5.4; 448

Displays the given top-level entry's description with pre and post hooks for the tree styles.

\glstreeDescLoc{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossaries−extra−stylemods v1.41+

§8.6.5.4; 449

Formats the top-level description (if set) and location list for the tree styles.

\glstreegroupheaderfmt{ ⟨*text*⟩ } glossary−tree v4.22+

§8.6.5.4; 446

Used to format the group title for the treegroup and indexgroup styles.

\glstreegroupheaderskip glossaries−extra−stylemods v1.42+ §8.6.5.4; 447

After group header skip for the treegroup and indexgroup styles.

\glstreegroupskip glossaries−extra−stylemods v1.41+ §8.6.5.4; 447

Group skip for the tree and index styles.

\glstreeitem glossary−tree v4.26+

Used to indent the top-level entries for the index styles.

\glstreenamefmt { ⟨*text*⟩ } glossary−tree v4.08+

Used to format the name for the tree and index styles.

\glstreenavigationfmt { ⟨*text*⟩ } glossary−tree v4.22+ §8.6.5.4; 446

Used to format the navigation element for styles like treehypergroup.

\glstreeNoDescSymbolPreLocation glossaries−extra−stylemods v1.42+ §8.6.5.4; 449

Inserted before the location list when there's no description or symbol for the tree styles.

\glstreenonamechilddesc { ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.31+ §8.6.5.4; 448

Displays the given child entry's description and post hook for the treenoname styles.

\glstreenonameChildDescLoc { ⟨*entry-label*⟩ }
glossaries−extra−stylemods v1.45+

Displays the given child entry's description and location list for the treenoname styles.

\glstreenonamedesc{⟨*entry-label*⟩} glossaries−extra−stylemods v1.31+ §8.6.5.4; 448

Displays the given top-level entry's description with pre and post hooks for the treenoname styles.

\glstreenonameDescLoc{⟨*entry-label*⟩} glossaries−extra−stylemods v1.45+

Displays the given top-level entry's description and location list for the treenoname styles.

\glstreenonamesymbol{⟨*entry-label*⟩} glossaries−extra−stylemods v1.31+ §8.6.5.4; 448

Displays the given top-level entry's symbol in parentheses for the treenoname styles.

\glstreepredesc glossary−tree v4.26+ §8.6.5.4; 447

Space inserted before top-level descriptions.

\glstreePreHeader{⟨*group-label*⟩}{⟨*group-title*⟩}
glossaries−extra−stylemods v1.41+ §8.6.5.4; 447

Pre group header hook the treegroup and indexgroup styles.

\glstreeprelocation *initial:* `\glsxtrprelocation`
glossaries−extra−stylemods v1.21+ §8.6.5.4; 448

Used before the top-level entry location list for the tree and index styles.

\glstreesubgroupitem`{previous group level}{level}{parent label}{group label}{group title}`
glossaries−extra−stylemods v1.49+

Used to display the sub-group header in the treegroup styles.

\glstreesubitem glossary−tree v4.26+

Used to indent the level 1 entries for the index styles.

\glstreeSubPreHeader { ⟨*previous group level*⟩ } { ⟨*level*⟩ } { ⟨*parent label*⟩ } { ⟨*group label*⟩ } { ⟨*group title*⟩ } glossaries−extra−stylemods v1.49+

Pre sub-group header hook the treegroup and indexgroup styles.

\glstreesubsubitem glossary−tree v4.26+

Used to indent the level 2 entries for the index styles.

\glstreesymbol { ⟨*entry-label*⟩ } glossaries−extra−stylemods v1.31+ §8.6.5.4; 449

Displays the top-level symbol in parentheses, if set, for the tree styles.

\glstriggerrecordformat { ⟨*location*⟩ } glossaries−extra v1.21+ §11.5; 571

Used as a special location format that indicates that the record is a trigger record.

\glsunexpandedfieldvalue { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } glossaries v4.48+

For use in expandable contexts where the field value is required but the contents should not be expanded. The field should be identified by its internal field label. Expands to nothing with no error or warning if the entry or field aren't defined.

\glsunset { ⟨*entry-label*⟩ } glossaries

Globally unsets the entry's first use flag. That is, this marks the entry as "used".

\glsunsetall [⟨*types*⟩] glossaries

Globally unsets all entries associated with the listed glossaries or all glossaries if ⟨*types*⟩ is omitted.

\glsunsetcategoryattribute { ⟨*category*⟩ } { ⟨*attribute*⟩ }
glossaries−extra v1.47+

§10.2.2; 536

Locally unsets the given attribute for the given category.

\glsupdatewidest [⟨*level*⟩] { ⟨*name*⟩ } glossaries−extra−stylemods v1.23+

§8.6.5.4; 450

Similar to `\glssetwidest` but only if ⟨*name*⟩ is wider than the current widest value for the given hierarchical level.

\glsuppercase { ⟨*text*⟩ } glossaries v4.50+

§5.2.3; 204

Converts ⟨*text*⟩ to uppercase.

\glsuseabbrvfont { ⟨*style-name*⟩ } { ⟨*text*⟩ } glossaries−extra v1.21+

§4.5.2; 165

Formats ⟨*text*⟩ according to the short format for the given abbreviation style.

\glsuselongfont { ⟨*style-name*⟩ } { ⟨*text*⟩ } glossaries−extra v1.21+

§4.5.2; 165

Formats ⟨*text*⟩ according to the long format for the given abbreviation style.

\glsuserdescription { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.30+

143

Description encapsulator for styles like long−short−user.

\GLSuseri [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As `\glsuseri` but converts the link text to all caps.

\Glsuseri [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As `\glsuseri` but converts the link text to sentence case.

\glsuseri [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `user1` value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options.

\glsuseriaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the `user1access` replacement text (if set).

\GLSuserii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As `\glsuserii` but converts the link text to all caps.

\Glsuserii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As `\glsuserii` but converts the link text to sentence case.

\glsuserii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `user2` value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options.

\glsuseriiaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the `user2access` replacement text (if set).

\GLSuseriii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As `\glsuseriii` but converts the link text to all caps.

\Glsuseriii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuseriii but converts the link text to sentence case.

\glsuseriii [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the user3 value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuseriiiaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the user3access replacement text (if set).

\GLSuseriv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuseriv but converts the link text to all caps.

\Glsuseriv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuseriv but converts the link text to sentence case.

\glsuseriv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the user4 value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuserivaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the user4access replacement text (if set).

\GLSuserv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuserv but converts the link text to all caps.

\Glsuserv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuserv but converts the link text to sentence case.

\glsuserv [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the user5 value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuservaccessdisplay { ⟨*text*⟩ } { ⟨*entry-label*⟩ } glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the user5access replacement text (if set).

\GLSuservi [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuservi but converts the link text to all caps.

\Glsuservi [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

As \glsuservi but converts the link text to sentence case.

\glsuservi [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries v2.04+

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the user6

value. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options.

`\glsuserviaccessdisplay`{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−accsupp v4.45+

Does ⟨*text*⟩ with the `user6access` replacement text (if set).

`\glswriteentry`{⟨*entry-label*⟩}{⟨*code*⟩} §5.8; 273

Performs the indexing code unless indexing should be suppressed.

`\glswriteglossarygroup`{⟨*group-label*⟩}{⟨*group-title*⟩} glossaries v5.1+

Does nothing by default, but the `saveglossarygroups` package option will redefine this command so that it saves the group information to the `aux` file.

`\glswriteglossarysubgroup`{⟨*level*⟩}{⟨*parent−label*⟩}{⟨*group−label*⟩}{⟨*group−title*⟩} glossaries v5.1+ §8.4.1; 405

Does nothing by default, but the `saveglossarygroups` package option will redefine this command so that it saves the sub-group information to the `aux` file.

`\glsxpabbrvfont`{⟨*text*⟩}{⟨*category*⟩} glossaries−extra v1.49+ §4.5.3.1; 173

If the `markshortwords` attribute is set for the given category, this encapsulates ⟨*text*⟩ with `\glsabbrvfont` otherwise with `\glsinnerfmtabbrvfont`. This command has to expand, so protect any content that shouldn't expand.

`\glsxplongfont`{⟨*text*⟩}{⟨*category*⟩} glossaries−extra v1.49+ §4.5.3.1; 173

If the `markwords` attribute is set for the given category, this encapsulates ⟨*text*⟩ with `\glslongfont` otherwise with `\glsinnerfmtlongfont`. This command has to expand, so protect any content that shouldn't expand.

Glsxtr

\Glsxtr [⟨*gls-options*⟩] [⟨*dfn-options*⟩] { ⟨*entry-label*⟩ }

§13; 629

As \glsxtr but applies sentence case.

\glsxtr [⟨*gls-options*⟩] [⟨*dfn-options*⟩] { ⟨*entry-label*⟩ }

§13; 628

If ⟨*entry-label*⟩ has already been defined, this just references it, otherwise the entry is defined. This command must be enabled with \GlsXtrEnableOnTheFly.

\glsxtr@counterrecord { ⟨*entry-label*⟩ } { ⟨*counter*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+

§8.4.3.2; 421

An aux file command used with \GlsXtrRecordCounter to append ⟨*value*⟩ to the record.⟨*counter*⟩ field. Also implements \glsxtrAddCounterRecordHook.

\glsxtr@record { ⟨*entry-label*⟩ } { ⟨*h-prefix*⟩ } { ⟨*counter*⟩ } { ⟨*encap*⟩ }
{ ⟨*location*⟩ } glossaries−extra v1.08+

§11.6.6; 604

Used in the aux file to provide the record for bib2gls (record=only). Ignored by LaTeX.

\glsxtr@record@nameref { ⟨*entry-label*⟩ } { ⟨*h-prefix*⟩ } { ⟨*counter*⟩ }
{ ⟨*encap*⟩ } { ⟨*location*⟩ } { ⟨*current title*⟩ } { ⟨*current anchor*⟩ } { ⟨*the-h-counter*⟩ }
glossaries−extra v1.37+

§11.6.6; 604

Used in the aux file to provide the nameref record for bib2gls. Ignored by LaTeX.

\glsxtr@resource { ⟨*options*⟩ } { ⟨*basename*⟩ } glossaries−extra v1.08+

§11; 542

Used in the aux file to provide the resource options for bib2gls for each resource set. Ignored by LaTeX.

\glsxtrabbreviationfont { ⟨*text*⟩ } glossaries−extra v1.30+

§5.5.2; 245

Used by \glsentryfmt to encapsulate non-regular entries the have the short field set.

\glsxtrabbrvfootnote{⟨*entry-label*⟩}{⟨*text*⟩} glossaries−extra v1.07+ 151

Command that produces the footnote for the footnote abbreviation styles, such as short−footnote and short−postfootnote.

\glsxtrabbrvpluralsuffix *initial:* \glspluralsuffix
glossaries−extra v1.12+ §4.1.2; 45

The default plural suffix used for abbreviations.

\glsxtrabbrvtype *initial:* \glsdefaulttype glossaries−extra §4.1.4; 45

Expands to the label of the default abbreviation glossary. The abbreviations package option will redefine this to abbreviations.

\glsxtrAccSuppAbbrSetFirstLongAttrs{⟨*category*⟩}
glossaries−extra v1.42+ §4.5.3.1; 174

Initialised accessibility support for the name, text and plural fields (if enabled with accsupp). This command is provided for abbreviation styles where the name and text are just the formatted abbreviation.

\glsxtrAccSuppAbbrSetNameLongAttrs{⟨*category*⟩}
glossaries−extra v1.42+ §4.5.3.1; 174

Initialised accessibility support for the first, firstplural, text and plural fields (if enabled with accsupp). This command is provided for abbreviation styles where the first and text are just the formatted abbreviation.

\glsxtrAccSuppAbbrSetNameShortAttrs{⟨*category*⟩}
glossaries−extra v1.42+ §4.5.3.1; 175

Initialised accessibility support for the name field (if enabled with accsupp). This command is provided for abbreviation styles where only the name is just the formatted abbreviation.

\glsxtrAccSuppAbbrSetNoLongAttrs{⟨*category*⟩} glossaries−extra v1.42+ §4.5.3.1; 174

Initialised accessibility support for the `name`, `first`, `firstplural`, `text` and `plural` fields (if enabled with `accsupp`). This command is provided for abbreviation styles where the `name`, `first` and `text` are just the formatted abbreviation.

\glsxtrAccSuppAbbrSetTextShortAttrs{⟨*category*⟩}
glossaries−extra v1.42+ §4.5.3.1; 174

Initialised accessibility support for the `text` and `plural` fields (if enabled with `accsupp`). This command is provided for abbreviation styles where the `text` is just the formatted abbreviation.

\glsxtractivatenopost glossaries−extra v1.22+ §8.5; 430

Activates `\nopostdesc` and `\glsxtrnopostpunc`.

\glsxtractualanchor glossaries−extra−bib2gls v1.49+ §11.6.6; 606

Expands to the anchor required by `\glsxtrdisplaylocnameref`.

\glsxtraddallcrossrefs §5.9.3; 288

Iterates over all defined entries and indexes any cross-references (identified by the `see` or `seealso` keys) that haven't been used.

\glsxtrAddCounterRecordHook{⟨*entry-label*⟩}{⟨*counter*⟩}{⟨*value*⟩}
glossaries−extra v1.49+ §8.4.3.2; 421

User-level hook used by `\glsxtr@counterrecord`. If this command is redefined, it must be done so in the preamble before the `aux` file is input.

\glsxtraddgroup{⟨*entry-label*⟩}{⟨*code*⟩} glossaries−extra v1.49+ §8.4.1; 403

Used by the "unsrt" family of commands to perform ⟨*code*⟩ if the entry identified by ⟨*entry-label*⟩ should have support for groups.

\glsxtraddlabelprefix{ ⟨*label-prefix*⟩ } glossaries−extra−bib2gls v1.37+ §11.6.7; 612

Appends ⟨*label-prefix*⟩ to the list of known labels.

\glsxtraddpunctuationmark{ ⟨*token(s)*⟩ } §5.5.4; 254

Adds ⟨*token(s)*⟩ to the list of punctuation characters used by `\glsxtrifnextpunc`. You may list multiple characters at the same time to add a batch, but don't add any separators (including spaces). Note that each character must be a single token, which means a single-byte character for pdfLaTeX. Multi-byte characters (UTF-8) will required a native Unicode engine (XeLaTeX or LuaLaTeX).

\glsxtraddunusedxrefs glossaries−extra v1.49+ §5.9.3; 288

Indexes any cross-references (identified by the `see` or `seealso` keys) that haven't been used.

\glsxtralias{ ⟨*entry-label*⟩ } glossaries−extra v1.12+ §5.9.2; 286

Expands to the value of the `alias` field for the entry identified by ⟨*entry-label*⟩. If the field isn't set, this will expand to nothing. If the entry isn't defined, this will expand to `\relax`.

\glsxtraliashook{ ⟨*entry-label*⟩ } glossaries−extra v1.49+ §3.4; 37

Hook implemented when the `alias` key is provided when an entry is defined.

\glsxtrAltTreeIndent glossaries−extra−stylemods v1.05+ §8.6.5.4; 453

Length register for the subsequent paragraph indentation for the alttree-like styles.

\glsxtralttreeInit glossaries−extra−stylemods v1.05+ §8.6.5.4; 453

Initialisation code performed by the alttree-like styles.

\glsxtralttreeSubSymbolDescLocation{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossaries−extra−stylemods v1.05+ §8.6.5.4; 452

Formats the symbol, description and location for child entries for the alttree-like styles.

\glsxtralttreeSymbolDescLocation{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossaries−extra−stylemods v1.05+ §8.6.5.4; 452

Formats the symbol, description and location for top-level entries for the alttree-like styles.

\glsxtrapptocsvfield{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*element*⟩ } glossaries−extra v1.47+ §3.5; 40

For use with fields that should contain comma-separated lists, this will append a command followed by ⟨*element*⟩ to the field value. If the field isn't defined, this command will behave like `\glsxtrdeffield`. No existence check is performed.

\GlsXtrAppToDefaultGlsOpts{ ⟨*options*⟩ } glossaries−extra v1.49+ §5.1.1; 191

Locally append ⟨*options*⟩ to the default options for the `\gls`-like and `\glstext`-like commands.

\glsxtrassignactualsetup glossaries−extra v1.42+ §9.1; 508
(requires `accsupp`)

Used to strip common formatting commands from a field value to supply the text-only accessibility content when initialising the default `shortaccess` and `shortpluralaccess` values.

\glsxtrassignfieldfont{ ⟨*entry-label*⟩ } glossaries−extra v1.04+ §5.5.2; 246

Used by the `\glstext`-like commands to initialise the formatting commands required for the given entry.

\glsxtrassignlinktextfmt glossaries−extra v1.49+ §5.5.4; 259

Initialised by the \gls-like and \glstext-like commands, this contains the definitions of \glslabel, \glstextformat, \glsxtrgenentrytextfmt.

\glsxtrattrentrytextfmt { ⟨*text*⟩ } glossaries−extra v1.49+ §5.5.3; 248

Applies the command obtained from the control sequence name supplied in the innertext-format attribute for the category assigned to the entry given by \glslabel. This command isn't used by default as it should rarely be needed an increases complexity.

\GlsXtrAutoAddOnFormat [⟨*entry-label*⟩] { ⟨*format list*⟩ } { ⟨*glsadd options*⟩ } glossaries−extra v1.37+ §5.8; 268

Identifies formats that should trigger an automatic \glsadd by the \gls-like and \glstext-like commands.

\glsxtrautoindex *initial:* \index glossaries−extra v1.16+ §12; 625

The indexing command used by by the auto-indexing feature.

\glsxtrautoindexassignsort { ⟨*entry-label*⟩ } glossaries−extra v1.16+ §12; 625

Used to assign the sort value for the auto-indexing feature.

\glsxtrautoindexentry { ⟨*entry-label*⟩ } glossaries−extra v1.16+ §12; 624

Expands to the "actual" part for the auto-indexing feature.

\glsxtrautoindexesc glossaries−extra v1.36+ §12; 625

Escapes the sort value used by the auto-indexing feature.

\glsxtrBasicDigitrules glossaries−extra−bib2gls v1.27+ 595

Expands to the Basic Latin digit character sort rules.

\glsxtrbibaddress{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the address field.

\glsxtrbibauthor{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the author field.

\glsxtrbibbooktitle{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the booktitle field.

\glsxtrbibchapter{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the chapter field.

\glsxtrbibedition{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the edition field.

\glsxtrbibhowpublished{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the howpublished field.

\glsxtrbibinstitution{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the institution field.

\glsxtrbibjournal{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the journal field.

\glsxtrbibmonth{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the month field.

\glsxtrbibnote{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the note field.

\glsxtrbibnumber{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the number field.

\glsxtrbiborganization{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the organization field.

\glsxtrbibpages{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the pages field.

\glsxtrbibpublisher{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the publisher field.

\glsxtrbibschool{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the school field.

\glsxtrbibseries{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the series field.

\GlsXtrBibTeXEntryAliases glossaries-extra-bib2gls v1.29+ §11.6.2; 583

Expands to the BibTeX to bib2gls entry aliases for use in entry-type-aliases.

\glsxtrbibtitle{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the title field.

\glsxtrbibtype{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the bibtextype field.

\glsxtrbibvolume{⟨*entry-label*⟩} (defined by \GlsXtrProvideBibTeXFields) §11.6.2; 583

Accesses the volume field.

\glsxtrbookindexatendgroup{⟨*entry-label*⟩} glossary−bookindex v1.21+ §8.7.1; 458

Used by the bookindex style at the end of a letter group (where the last top-level entry is given by ⟨*entry-label*⟩).

\glsxtrbookindexbetween{⟨*entry1-label*⟩}{⟨*entry2-label*⟩}
glossary−bookindex v1.21+ §8.7.1; 457

Used by the bookindex style between two entries where ⟨*entry1-label*⟩ is the last top-level entry and ⟨*entry2-label*⟩ is the next entry, which is a top-level entry.

\glsxtrbookindexbookmark{⟨*group-title*⟩}{⟨*bookmark-name*⟩}
glossary−bookindex v1.21+ §8.7.1; 459

Adds a bookmark with `\pdfbookmark`, if supported.

\glsxtrbookindexcols *initial:* 2 glossary−bookindex v1.21+ §8.7.1; 455

Expands to the number of columns for the bookindex style.

\glsxtrbookindexcolspread glossary−bookindex v1.12+ §8.7.1; 455

If not empty this should expand to the option argument for multicols.

\glsxtrbookindexfirstmark glossary−bookindex v1.21+ §8.7.1; 460

Used by the bookindex style to obtain the first mark and, if found, format it with `\glsxtrbookindexfirstmarkfmt`.

\glsxtrbookindexfirstmarkfmt{⟨*entry-label*⟩} glossary−bookindex v1.21+ §8.7.1; 460

Used by the bookindex style to format the first mark.

\glsxtrbookindexformatheader{ ⟨*group-title*⟩ } glossary−bookindex v1.21+ §8.7.1; 459

Used by the bookindex style to format a group header.

\glsxtrbookindexformatsubheader{ ⟨*previous level*⟩ } { ⟨*level*⟩ } { ⟨*parent-label*⟩ } { ⟨*group-label*⟩ } { ⟨*title*⟩ } glossary−bookindex v1.49+

Formats the sub-group header.

\glsxtrbookindexlastmark glossary−bookindex v1.21+ §8.7.1; 460

Used by the bookindex style to obtain the last mark and, if found, format it with `\glsxtrbookindexlastmarkfmt`.

\glsxtrbookindexlastmarkfmt{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 460

Used by the bookindex style to format the last mark.

\glsxtrbookindexlocation{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ } glossary−bookindex v1.39+ §8.7.1; 457

Used by the bookindex style to display top-level location lists.

\glsxtrbookindexmarkentry{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 460

Used by the bookindex style to mark an entry in the `aux` file.

\glsxtrbookindexmulticolsenv glossary−bookindex v1.25+ §8.7.1; 455

Expands to the name of the multicols environment to use.

\glsxtrbookindexname{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 455

Used by the bookindex style to display a top-level entry's name.

\glsxtrbookindexparentchildsep glossary−bookindex v1.21+ §8.7.1; 457

Used by the bookindex style to separate a top-level parent and child entry.

\glsxtrbookindexparentsubchildsep glossary−bookindex v1.21+ §8.7.1; 457

Used by the bookindex style to separate a sub-level parent and child entry.

\glsxtrbookindexpregroupskip{ ⟨*skip*⟩ } glossary−bookindex v1.49+ §8.7.1; 459

Used by the bookindex style insert ⟨*skip*⟩ after a group header.

\glsxtrbookindexprelocation{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 456

Used by the bookindex style to display a separator before top-level location lists.

\glsxtrbookindexsubatendgroup{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 458

Used by the bookindex style at the end of a letter group (where the last level 1 entry is given by ⟨*entry-label*⟩).

\glsxtrbookindexsubbetween{ ⟨*entry1-label*⟩ } { ⟨*entry2-label*⟩ }
glossary−bookindex v1.21+ §8.7.1; 457

As `\glsxtrbookindexbetween` but for level 1 entries.

\glsxtrbookindexsublocation{ ⟨*entry-label*⟩ } { ⟨*location list*⟩ }
glossary−bookindex v1.39+ §8.7.1; 457

Used by the bookindex style to display child location lists.

\glsxtrbookindexsubname{ ⟨*entry-label*⟩ } glossary−bookindex v1.21+ §8.7.1; 456

Used by the bookindex style to display a child entry's name.

\glsxtrbookindexsubprelocation{ ⟨*entry-label*⟩ }
glossary−bookindex v1.21+

§8.7.1; 457

Used by the bookindex style to display a separator before child location lists.

\glsxtrbookindexsubsubatendgroup{ ⟨*entry-label*⟩ }
glossary−bookindex v1.21+

§8.7.1; 458

Used by the bookindex style at the end of a letter group (where the last level 2 entry is given by ⟨*entry-label*⟩).

\glsxtrbookindexsubsubbetween{ ⟨*entry1-label*⟩ } { ⟨*entry2-label*⟩ }
glossary−bookindex v1.21+

§8.7.1; 457

As `\glsxtrbookindexbetween` but for level 2 entries.

\glsxtrbookindexsubsubitem{ ⟨*level*⟩ } glossary−bookindex v1.54+

§8.7.1; 459

Used to start sub-sub items and lower. The ⟨*level*⟩ will be 2 or more.

\glsxtrbookindexsubtarget{ ⟨*entry-label*⟩ } { ⟨*text*⟩ }
glossary−bookindex v1.54+

§8.7.1; 455

Used by the bookindex style to create the target for child items.

\glsxtrbookindextarget{ ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossary−bookindex v1.54+

§8.7.1; 455

Used by the bookindex style to create the target for top-level items.

\glsxtrcat *initial:* `general`

§13; 630

Expands to the default category set by commands like `\glsxtr`.

\glsxtr⟨*category*⟩⟨*field*⟩**accsupp** (user defined)

Expands to the accessibility support command for the given internal field label and category, which is used by `\glsfieldaccsupp`.

\glsxtrchecknohyperfirst{⟨*entry-label*⟩} glossaries−extra v1.07+ §5.1.1; 193

Sets `hyper=false` if the `nohyperfirst` attribute is set.

\GlsXtrClearAutoAddOnFormat glossaries−extra v1.59+ §5.8; 269

Clears the formats that should trigger an automatic `\glsadd`.

\glsxtrclearlabelprefixes glossaries−extra−bib2gls v1.37+ §11.6.7; 612

Clears the list of known prefixes.

\GlsXtrClearUnsetBuffer glossaries−extra v1.49+ §5.10.1; 293

Locally clears the buffer, but doesn't stop buffering.

\glsxtrcombiningdiacriticIIIrules glossaries−extra−bib2gls v1.27+ 591

Expands to the third set of combining diacritic sort rules.

\glsxtrcombiningdiacriticIIrules glossaries−extra−bib2gls v1.27+ 591

Expands to the second set of combining diacritic sort rules.

\glsxtrcombiningdiacriticIrules glossaries−extra−bib2gls v1.27+ 590

Expands to the first set of combining diacritic sort rules.

\glsxtrcombiningdiacriticIVrules glossaries−extra−bib2gls v1.27+ 591

Expands to the fourth set of combining diacritic sort rules.

\glsxtrcombiningdiacriticrules glossaries−extra−bib2gls v1.27+ 590

Expands to all the combining diacritic sort rules.

\glsxtrcontrolIIrules glossaries−extra−bib2gls v1.54+ 590

Expands to a subset of ordered control character sort rules (information separators).

\glsxtrcontrolIrules glossaries−extra−bib2gls v1.54+ 590

Expands to a subset of equivalent control character sort rules.

\glsxtrcontrolrules glossaries−extra−bib2gls v1.27+ 589

Expands to control character sort rules.

\glsxtrcopytoglossary { ⟨*entry-label*⟩ } { ⟨*glossary-type*⟩ } *modifier:* * glossaries−extra v1.12+ §8; 384

Copies the entry to the internal glossary list for the given glossary. The starred version performs a global change. The unstarred version can be localised. Only for use with the "unsrt" family of commands.

\glsxtr⟨*counter*⟩**locfmt** { ⟨*location*⟩ } { ⟨*title*⟩ } (user defined) §11.6.6; 607

Used by `\glsxtrdisplaylocnameref` for format a location where the counter matches ⟨*counter*⟩.

\glsxtrcurrencyrules glossaries−extra−bib2gls v1.27+ 594

Expands to currency character sort rules.

\glsxtrcurrentfield glossaries−extra v1.49+ §5.5.4; 257

Placeholder command for use in post-link hooks. This expands to empty if the calling command was one of the \gls-like commands or it was one of the inline full form commands, otherwise it will expand to the name of the key associated with the *singular* form of the command.

\glsxtrcurrentmglscsname glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the control sequence name of the calling command.

\glsxtrdefaultentrytextfmt { ⟨*text*⟩ } glossaries−extra v1.49+ §5.5.3; 248

Default inner formatting. Initialised to just do ⟨*text*⟩.

\GlsXtrDefaultResourceOptions glossaries−extra v1.40+ §11; 543

Expands to default resource options.

\glsxtrdefaultrevert { ⟨*text*⟩ } glossaries−extra v1.49+ 139

The default definition of \glsxtrrevert. Simply does ⟨*text*⟩.

\GLSxtrdefaultsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 179
glossaries−extra v1.49+

The default all caps subsequent format style that only shows the short form and insert (with support for innertextformat).

\Glsxtrdefaultsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 179
glossaries−extra v1.17+

The default sentence case subsequent format style that only shows the short form and insert (with support for innertextformat).

\glsxtrdefaultsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.17+ 179

The default subsequent format style that only shows the short form and insert (with support for `innertextformat`).

\GLSxtrdefaultsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.49+ 179

The default all caps subsequent plural format style that only shows the short form and insert (with support for `innertextformat`).

\Glsxtrdefaultsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.17+ 179

The default sentence case subsequent plural format style that only shows the short form and insert (with support for `innertextformat`).

\glsxtrdefaultsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.17+ 179

The default subsequent plural format style that only shows the short form and insert (with support for `innertextformat`).

\glsxtrdeffield { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*value*⟩ } glossaries−extra v1.12+ §3.5; 39

Like `\GlsXtrSetField` but doesn't perform any existence checks.

\GlsXtrDefineAbbreviationShortcuts

Used by `shortcuts=abbreviations` and `shortcuts=all`. This command redefines itself to do nothing because it can only be used once.

\GlsXtrDefineAcronymShortcuts glossaries−extra v1.17+

Used by `shortcuts=ac` and `shortcuts=acother`. This command redefines itself to do nothing because it can only be used once.

\GlsXtrDefineOtherShortcuts

Used by `shortcuts=other` and `shortcuts=all`. This command redefines itself to do nothing because it can only be used once.

\glsxtrdetoklocation{ ⟨*location*⟩ } glossaries−extra v1.21+ §11.5; 568

Just expands to ⟨*location*⟩ by default but may be redefined to help protect awkward characters.

\glsxtrdigitrules glossaries−extra−bib2gls v1.27+ 594

Expands to 0–9 digit character sort rules (includes superscript and subscript digits).

\glsxtrdiscardperiod{ ⟨*entry-label*⟩ } { ⟨*discarded*⟩ } { ⟨*no discard*⟩ } ⟨*token*⟩
glossaries−extra §5.5.4; 252

If ⟨*token*⟩ is a full stop and the entry's category attributes indicate that a full stop should be discarded (such as `discardperiod`), then ⟨*discarded*⟩ is performed, otherwise ⟨*no discard*⟩ is done and the ⟨*token*⟩ is processed. The actual test to determine if ⟨*token*⟩ is a full stop is performed by `\glsxtrifperiod`. This command is used in post-link hooks.

\glsxtrdiscardperiodretainfirstuse{ ⟨*entry-label*⟩ } { ⟨*discarded*⟩ } { ⟨*no discard*⟩ } ⟨*token*⟩ glossaries−extra v1.49+ §5.5.4; 252

Used to discard a following full stop when the `retainfirstuseperiod` attribute is set.

\GlsXtrDiscardUnsetBuffering glossaries−extra v1.42+ §5.10.1; 294

Discards the pending buffer and restores `\glsunset`.

\glsxtrdisplayendloc{ ⟨*format*⟩ } { ⟨*location*⟩ } glossaries−extra v1.12+ §8.6.3; 441

Used to display an end location from an explicit range.

\glsxtrdisplayendlochook glossaries−extra v1.12+ §8.6.3; 441

Hook used by \glsxtrdisplayendloc.

\glsxtrdisplaylocnameref{⟨*prefix*⟩}{⟨*counter*⟩}{⟨*format*⟩}{⟨*location*⟩}{⟨*title*⟩}{⟨*href*⟩}{⟨*hcounter*⟩}{⟨*file*⟩} glossaries−extra−bib2gls v1.37+ §11.6.6; 605

Used to display records created with record=nameref.

\glsxtrdisplaysingleloc{⟨*format*⟩}{⟨*location*⟩} glossaries−extra v1.12+ §8.6.3; 441

Used to display a single location.

\glsxtrdisplaystartloc{⟨*format*⟩}{⟨*location*⟩} glossaries−extra v1.12+ §8.6.3; 441

Used to display a start location from an explicit range.

\glsxtrdisplaysupploc{⟨*prefix*⟩}{⟨*counter*⟩}{⟨*format*⟩}{⟨*src*⟩}{⟨*location*⟩} *modifier:* * glossaries−extra−bib2gls v1.36+ §11.6.5; 604

Like \glsnoidxdisplayloc but used for supplementary locations.

\glsxtrdoautoindexname{⟨*entry-label*⟩}{⟨*attribute*⟩} §12; 624

Used to automatically index (using \glsxtrautoindex) the entry's name, if the given attribute is set for the entry's category.

\glsxtrdopostpunc{⟨*code*⟩}⟨*token*⟩ glossaries−extra v1.49+ §5.5.4; 254

If ⟨*token*⟩ is a recognised punctuation character (see \glsxtrifnextpunc) this does the punctuation character and then ⟨*code*⟩, otherwise if does ⟨*code*⟩ followed by ⟨*token*⟩.

\glsxtrdowrglossaryhook{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.8; 272

Hook used whenever an entry is indexed. Does nothing by default.

\GlsXtrDualBackLink{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−extra−bib2gls v1.30+ §11.6.7; 610

Adds a hyperlink to the given entry's dual (whose label is stored in the field given by \GlsXtrDualField) with the given hyperlink text.

\GlsXtrDualField *initial:* dual glossaries−extra−bib2gls v1.30+ §11.6.7; 611

Expands to the internal field label used by \GlsXtrDualBackLink.

\glsxtredeffield{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩}
glossaries−extra v1.12+ §3.5; 39

Like \glsxtrdeffield but (protected) expands ⟨*value*⟩.

\glsxtremrevert{⟨*text*⟩} glossaries−extra v1.49+ 164

The definition of \glsxtrrevert used by the emphasized ("em") abbreviation styles. Uses \textup.

\glsxtremsuffix *initial:* \glsxtrabbrvpluralsuffix 164

The plural suffix used by the emphasized ("em") abbreviation styles.

\GlsXtrEnableEntryCounting{⟨*category-list*⟩}{⟨*trigger-value*⟩} §6.1; 318

Enables entry counting for the given list of categories with the given trigger value (which must be an integer).

\GlsXtrEnableEntryUnitCounting{⟨*category-list*⟩}{⟨*trigger-value*⟩}
{⟨*counter*⟩} §6.1; 323

Enables unit entry counting for the given list of categories with the given trigger value (which must be an integer) and the associated counter.

\GlsXtrEnableIndexFormatOverride (preamble only)

§12; 626

Allows the `format` key to override the attribute value.

\GlsXtrEnableInitialTagging{ ⟨*categories*⟩ } { ⟨*cs*⟩ } *modifier:* *

§4.4; 57

Robustly defines the command ⟨*cs*⟩ to accept a single argument, which is a letter (or letters) that needs to be tagged. The unstarred version triggers an error if ⟨*cs*⟩ is already defined. The unstarred version will redefine ⟨*cs*⟩ if it already exists.

\GlsXtrEnableLinkCounting [⟨*parent counter*⟩] { ⟨*categories*⟩ }
glossaries−extra v1.26+
(preamble only)

§6.2; 327

Enables link counting for the given categories.

\GlsXtrEnableOnTheFly *modifier:* *

§13; 628

Enables on the fly commands, such as `\glsxtr`.

\GlsXtrEnablePreLocationTag{ ⟨*page tag*⟩ } { ⟨*pages tag*⟩ }
glossaries−extra v1.04+

§8.6.3; 440

Enables the location list tag.

\glsxtrenablerecordcount glossaries−extra v1.21+

§11.5; 573

Redefines the `\gls`-like commands (except `\glsdisp`) to use the analogous record count commands (`\rgls` etc).

\glsxtrendfor glossaries−extra v1.24+

§5.13; 310

When used within `\glsxtrforcsvfield` signifies that the loop should break at the end of the current iteration.

\Glsxtrentryfmt { ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossaries−extra v1.49+ §5.12.2; 307

As \glsxtrentryfmt but converts ⟨*text*⟩ to sentence case.

\glsxtrentryfmt { ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossaries−extra v1.12+ §5.12.2; 304

Does \⟨*csname*⟩ { ⟨*text*⟩ } where the control sequence name ⟨*csname*⟩ is obtained from the field given by \GlsXtrFmtField. If hyperref has been loaded and this command will expand to \glsxtrpdfentryfmt{⟨*entry-label*⟩}{⟨*text*⟩} in a PDF bookmark.

\glsxtrentryparentname { ⟨*entry-name*⟩ } glossaries−extra v1.39+ §5.11; 300

Expands to the name field of the given entry's parent or does nothing if the entry doesn't have the parent field set or isn't defined.

\glsxtrequationlocfmt { ⟨*location*⟩ } { ⟨*title*⟩ } glossaries−extra−bib2gls v1.42+ §11.6.6; 607

Used by \glsxtrdisplaylocnameref to format a location where the counter is equation.

\GlsXtrExpandedFmt { ⟨*cs*⟩ } { ⟨*content*⟩ } glossaries−extra v1.30+ §5.5; 241

Fully-expands ⟨*content*⟩ and passes it to ⟨*cs*⟩, which must be a command that takes a single argument.

\glsxtrfielddolistloop { ⟨*entry-label*⟩ } { ⟨*field*⟩ } glossaries−extra v1.12+ §5.14; 312

Iterates over the given field's value using etoolbox's \dolistcsloop.

\glsxtrfieldforlistloop { ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*handler-cs*⟩ }
glossaries−extra v1.12+ §5.14; 312

Iterates over the given field's value using etoolbox's \forlistcsloop.

\glsxtrfieldformatcsvlist { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.42+

§5.13; 310

Formats the comma-separated list stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ using datatool−base's `\DTLformatlist`. This command uses `\glsxtrifhasfield` so the complete list can be obtained with `\glscurrentfieldvalue`. This adds implicit grouping. There is no starred version.

\glsxtrfieldformatlist { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.42+

§5.14; 312

Formats the value of the given field, which should be an etoolbox internal list, using the same list handler macro as datatool's `\DTLformatlist`.

\glsxtrfieldifinlist { ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*item*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.12+

§5.14; 312

Uses etoolbox's `\ifinlistcs` to determine if ⟨*item*⟩ is in the list stored in the given field.

\glsxtrfieldlistadd { ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+

§3.5; 40

Appends ⟨*value*⟩ to the given field using etoolbox's `\listcsadd`.

\glsxtrfieldlisteadd { ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+

§3.5; 40

Appends ⟨*value*⟩ to the given field using etoolbox's `\listcseadd`.

\glsxtrfieldlistgadd { ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+

§3.5; 40

Appends ⟨*value*⟩ to the given field using etoolbox's `\listcsgadd`.

\glsxtrfieldlistxadd{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+

§3.5; 40

Appends ⟨*value*⟩ to the given field using etoolbox's `\listcsxadd`.

\glsxtrfieldtitlecase{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }

§5.11; 300

As `\glsxtrusefield` but converts the field value to title case.

\glsxtrfieldtitlecasecs{ ⟨*content*⟩ } glossaries−extra v1.07+

§5.11; 300

Converts ⟨*content*⟩ to title case (expanding the first token once). Uses `\glscapitalise-words`, if defined, otherwise uses `\capitalisewords`.

\glsxtrfieldxifinlist{ ⟨*entry-label*⟩ } { ⟨*field*⟩ } { ⟨*item*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.12+

§5.14; 313

Uses etoolbox's `\xifinlistcs` to determine if ⟨*item*⟩ is in the list stored in the given field.

\glsxtrfirstscfont{ ⟨*text*⟩ } glossaries−extra v1.04+

Maintained for backwards-compatibility used to typeset ⟨*text*⟩ in small capitals (`\textsc`) for the "sc" abbreviation styles on first use.

\glsxtrfirstsmfont{ ⟨*text*⟩ } glossaries−extra v1.04+

Maintained for backwards-compatibility used to typeset ⟨*text*⟩ in a smaller font (`\textsmaller`) for the "sm" abbreviation styles on first use.

\Glsxtrfmt [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossaries−extra v1.49+

§5.12.2; 306

As `\glsxtrfmt` but applies a sentence case change to ⟨*text*⟩.

\glsxtrfmt [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*text*⟩} glossaries−extra v1.12+ §5.12.2; 304

Behaves like \glslink [⟨*options*⟩] {⟨*entry-label*⟩} {\⟨*csname*⟩ {⟨*text*⟩}⟨*insert*⟩} where the control sequence name ⟨*csname*⟩ is obtained from the field given by \GlsXtrFmtField. The actual format of the link text is governed by \glsxtrfmtdisplay.

\Glsxtrfmt* [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*text*⟩} [⟨*insert*⟩] glossaries−extra v1.49+ §5.12.2; 306

As \glsxtrfmt* but applies a sentence case change to ⟨*text*⟩.

\glsxtrfmt* [⟨*options*⟩] {⟨*entry-label*⟩} {⟨*text*⟩} [⟨*insert*⟩] glossaries−extra v1.23+

As the unstarred version \glsxtrfmt but accepts the final ⟨*insert*⟩ option.

\GlsXtrFmtDefaultOptions *initial:* noindex glossaries−extra v1.12+ §5.1.1; 192

Expands to the default options for \glsxtrfmt.

\glsxtrfmtdisplay {⟨*csname*⟩} {⟨*text*⟩} {⟨*insert*⟩} glossaries−extra v1.23+ §5.12.2; 304

Formats the link text used in \glsxtrfmt.

\glsxtrfmtexternalnameref {⟨*target*⟩} {⟨*format*⟩} {⟨*title*⟩} {⟨*file*⟩}
glossaries−extra−bib2gls v1.37+ §11.6.6; 608

Used by \glsxtrnamereflink to create an external location hyperlink.

\GlsXtrFmtField *initial:* useri glossaries−extra v1.12+ §5.12.2; 304

Expands to the name of the used by \glsxtrfmt.

\glsxtrfmtinternalnameref {⟨*target*⟩} {⟨*format*⟩} {⟨*file*⟩}
glossaries−extra−bib2gls v1.37+ §11.6.6; 608

Used by \glsxtrnamereflink to create an internal location hyperlink.

\glsxtrfootnotedescname glossaries−extra v1.42+ 151

Expands to the name value for styles like short−footnote−desc.

\glsxtrfootnotedescsort glossaries−extra v1.42+ 151

Expands to the sort value for styles like short−footnote−desc.

\glsxtrfootnotelongformat { ⟨*entry-label*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 152

Used in the footnote text to format the singular long form.

\glsxtrfootnotelongplformat { ⟨*entry-label*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 152

Used in the footnote text to format the plural long form.

\glsxtrfootnotename glossaries−extra v1.25+ 150

Expands to the name value for styles like short−footnote.

\glsxtrforcsvfield { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*handler cs*⟩ }
modifier: * glossaries−extra v1.24+ §5.13; 309

Iterates over the comma-separated list stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ and performs ⟨*handler cs*⟩ { ⟨*element*⟩ } for each element of the list. This command uses `\glsxtrifhasfield` so the complete list can be obtained with `\glscurrentfieldvalue`. The unstarred version adds implicit grouping. The starred version doesn't.

\GlsXtrForeignText { ⟨*entry-label*⟩ } { ⟨*text*⟩ } glossaries−extra v1.32+ §5.12.1; 302

If the entry given by ⟨*entry-label*⟩ has the field identified by `\GlsXtrForeignTextField` then ⟨*text*⟩ will be encapsulated according to the language tag stored in that field (using tracklang's interface).

\GlsXtrForeignTextField glossaries−extra v1.32+ §5.12.1; 302

Expands to the internal field label used by \GlsXtrForeignText.

\GlsXtrFormatLocationList { ⟨*location list*⟩ } §8.6.3; 439

Used by \glossaryentrynumbers to encapsulate the entire location list in the glossary.

\GlsXtrForUnsetBufferedList { ⟨*handler-cs*⟩ } glossaries−extra v1.31+ §5.10.1; 294

Iterates over the labels stored in the current buffer.

\glsxtrfractionrules glossaries−extra−bib2gls v1.27+ 595

Expands to the number forms fraction character sort rules.

\GLSxtrfull [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrfull but converts the link text to all caps.

\Glsxtrfull [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrfull but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glsxtrfull [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the short and long values, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. The format produced by this command may not match the format produced by the first use of \gls { ⟨*entry-label*⟩ }, depending on the abbreviation style. For the first optional argument, see \glslink options.

\GLSxtrfullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 178

The all caps singular display full form (defined by the abbreviation style).

\Glsxtrfullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 178

The sentence case singular display full form (defined by the abbreviation style).

\glsxtrfullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 177

The singular display full form (defined by the abbreviation style).

\GLSxtrfullpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 55

As `\glsxtrfullpl` but converts the link text to all caps.

\Glsxtrfullpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 55

As `\glsxtrfullpl` but converts the first character of the link text to uppercase (for the start of a sentence) using `\makefirstuc`.

\glsxtrfullpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 55

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `short-plural` and `longplural` values, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. The format produced by this command may not match the format produced by the first use of `\glspl`{⟨*entry-label*⟩}, depending on the abbreviation style. For the first optional argument, see `\glslink` options.

\GLSxtrfullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 178

The all caps plural display full form (defined by the abbreviation style).

\Glsxtrfullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 178

The sentence case plural display full form (defined by the abbreviation style).

\glsxtrfullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 178

The plural display full form (defined by the abbreviation style).

\glsxtrfullsaveinsert { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ §4.3; 55

Implemented at the start of all the inline full form commands like `\glsxtrfull` to save the `\glsinsert` placeholder. By default, this just does `\glsxtrsaveinsert`.

\glsxtrfullsep { ⟨*entry-label*⟩ } 139

Separator used by the parenthetical inline full form and also for some display full forms.

\glsxtrgenabbrvfmt §5.5.5; 261

The display format used by `\glsentryfmt` for entries that have the `short` field set and have the `regular` attribute set to `false`.

\glsxtrgenentrytextfmt glossaries−extra v1.49+ §5.5.3; 248

Redefined by the `\gls`-like and `\glstext`-like hooks to set up the inner formatting. Initialised to `\glsxtrdefaultentrytextfmt`.

\glsxtrGeneralInitRules glossaries−extra−bib2gls v1.49+ 595

A shortcut that expands to the ignorable rules, combining diacritic rules, hyphen rules, general punctuation rules, digit rules, and fraction rules.

\glsxtrGeneralLatinAtoGrules glossaries−extra−bib2gls v1.49+ 597

Expands to the A–G subset of General Latin I sort rules.

\glsxtrGeneralLatinAtoMrules glossaries-extra-bib2gls v1.49+ 597

Expands to the A–M subset of General Latin I sort rules.

\glsxtrGeneralLatinHtoMrules glossaries-extra-bib2gls v1.49+ 597

Expands to the H–M subset of General Latin I sort rules.

\glsxtrGeneralLatinIIIrules glossaries-extra-bib2gls v1.27+ 596

Expands to the third set of General Latin sort rules.

\glsxtrGeneralLatinIIrules glossaries-extra-bib2gls v1.27+ 596

Expands to the second set of General Latin sort rules.

\glsxtrGeneralLatinIrules glossaries-extra-bib2gls v1.27+ 596

Expands to the first set of General Latin sort rules.

\glsxtrGeneralLatinIVrules glossaries-extra-bib2gls v1.27+ 596

Expands to the fourth set of General Latin sort rules.

\glsxtrGeneralLatinNtoSrules glossaries-extra-bib2gls v1.49+ 598

Expands to the N–S subset of General Latin I sort rules.

\glsxtrGeneralLatinNtoZrules glossaries-extra-bib2gls v1.49+ 597

Expands to the N–Z subset of General Latin I sort rules.

\glsxtrGeneralLatinTtoZrules glossaries-extra-bib2gls v1.49+ 598

Expands to the T–Z subset of General Latin I sort rules.

\glsxtrGeneralLatinVIIIrules glossaries-extra-bib2gls v1.27+ 597

Expands to the eighth set of General Latin sort rules.

\glsxtrGeneralLatinVIIrules glossaries-extra-bib2gls v1.27+ 597

Expands to the seventh set of General Latin sort rules.

\glsxtrGeneralLatinVIrules glossaries-extra-bib2gls v1.27+ 597

Expands to the sixth set of General Latin sort rules.

\glsxtrGeneralLatinVrules glossaries-extra-bib2gls v1.27+ 597

Expands to the fifth set of General Latin sort rules.

\glsxtrgeneralpuncaccentsrules glossaries-extra-bib2gls v1.49+ 593

Punctuation accent subset of `\glsxtrgeneralpuncIrules`.

\glsxtrgeneralpuncbracketIIIrules glossaries-extra-bib2gls v1.56+ 594

Third punctuation bracket subset (extra mathematical brackets).

\glsxtrgeneralpuncbracketIIrules glossaries-extra-bib2gls v1.56+ 594

Second punctuation bracket subset (mathematical brackets).

\glsxtrgeneralpuncbracketIrules glossaries-extra-bib2gls v1.56+ 593

First punctuation bracket subset.

\glsxtrgeneralpuncbracketIVrules glossaries-extra-bib2gls v1.56+ 594

Fourth punctuation bracket subset (ornamental brackets).

\glsxtrgeneralpuncbracketrules glossaries-extra-bib2gls v1.49+ 593

Punctuation bracket subset of `\glsxtrgeneralpuncIrules`.

\glsxtrgeneralpuncdotrules glossaries-extra-bib2gls v1.56+

Punctuation dot marks subset.

\glsxtrgeneralpuncIIIrules glossaries-extra-bib2gls v1.56+ 594

Expands to the third set of general punctuation sort rules (includes `\glsxtrminusrules` before plus).

\glsxtrgeneralpuncIIrules glossaries-extra-bib2gls v1.27+ 594

Expands to the second set of general punctuation sort rules.

\glsxtrgeneralpuncIrules glossaries-extra-bib2gls v1.27+ 592

Expands to the first set of general punctuation (including currency) sort rules.

\glsxtrgeneralpuncmarksrules glossaries-extra-bib2gls v1.49+ 593

Punctuation mark subset of `\glsxtrgeneralpuncIrules`.

\glsxtrgeneralpuncquoterules glossaries-extra-bib2gls v1.49+ 593

Punctuation quote subset of `\glsxtrgeneralpuncIrules`.

\glsxtrGeneralPuncRules glossaries-extra-bib2gls v1.56+ 596

A shortcut that expands to common punctuation rules, currency rules, digit rules, and fraction rules.

\glsxtrgeneralpuncrules glossaries-extra-bib2gls v1.27+ 592

Expands to all sets of general punctuation sort rules.

\glsxtrgeneralpuncsignrules glossaries-extra-bib2gls v1.49+ 594

Punctuation sign subset of `\glsxtrgeneralpuncIrules`.

\glsxtrgetgrouptitle{⟨*group-label*⟩}{⟨*cs*⟩} glossaries-extra v1.14+ §8.6.4; 443

Obtains the title corresponding to the group identified by ⟨*group-label*⟩ and stores the result in the control sequence ⟨*cs*⟩.

\glsxtrgroupfield *initial:* `group` glossaries-extra v1.21+ §8.4.1; 399

Expands to the internal field label used to store the group label (requires `record`).

\Glsxtrglossentry{⟨*entry-label*⟩} glossaries-extra v1.54+ §8.5; 428

As `\glsxtrglossentry` but applies sentence case.

\glsxtrglossentry{⟨*entry-label*⟩} glossaries-extra v1.21+ §8.5; 428

Used for standalone entries to display the name with `\glossentryname`, with appropriate hooks.

\Glsxtrglossentryother{⟨*header*⟩}{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.54+

§8.5; 430

As \glsxtrglossentryother but applies sentence case.

\glsxtrglossentryother{⟨*header*⟩}{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.22+

§8.5; 430

Like \glsxtrglossentry but uses the given field instead of name.

\GLSxtrheadfirst{⟨*entry-label*⟩} glossaries−extra v1.49+

§5.3.3; 229

Used to display the all caps entry's first field in the page header.

\Glsxtrheadfirst{⟨*entry-label*⟩}

§5.3.3; 229

Used to display the sentence case entry's first field in the page header (converts to all caps if headuc attribute is true).

\glsxtrheadfirst{⟨*entry-label*⟩}

§5.3.3; 229

Used to display the entry's first field in the page header (converts to all caps if headuc attribute is true).

\GLSxtrheadfirstplural{⟨*entry-label*⟩} glossaries−extra v1.49+

§5.3.3; 230

Used to display the all caps entry's firstplural field in the page header.

\Glsxtrheadfirstplural{⟨*entry-label*⟩}

§5.3.3; 230

Used to display the sentence case entry's firstplural field in the page header (converts to all caps if headuc attribute is true).

\glsxtrheadfirstplural{⟨*entry-label*⟩}

§5.3.3; 229

Used to display the entry's `firstplural` field in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSxtrheadfull{⟨*entry-label*⟩} glossaries−extra v1.49+

§5.3.3; 226

Used to display the entry's all caps full form in the page header.

\Glsxtrheadfull{⟨*entry-label*⟩} glossaries−extra v1.02+

§5.3.3; 226

Used to display the entry's sentence case full form in the page header (converts to all caps if `headuc` attribute is `true`).

\glsxtrheadfull{⟨*entry-label*⟩} glossaries−extra v1.02+

§5.3.3; 225

Used to display the entry's full form in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSxtrheadfullpl{⟨*entry-label*⟩} glossaries−extra v1.49+

§5.3.3; 226

Used to display the entry's all caps full plural form in the page header.

\Glsxtrheadfullpl{⟨*entry-label*⟩} glossaries−extra v1.02+

§5.3.3; 226

Used to display the entry's sentence case full plural form in the page header (converts to all caps if `headuc` attribute is `true`).

\glsxtrheadfullpl{⟨*entry-label*⟩} glossaries−extra v1.02+

§5.3.3; 226

Used to display the entry's full plural form in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSxtrheadlong{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 224

The behaviour of \GLSfmtlong when it occurs in a page header.

\Glsxtrheadlong{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 224

The behaviour of \Glsfmtlong when it occurs in a page header.

\glsxtrheadlong{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 224

The behaviour of \glsfmtlong when it occurs in a page header.

\GLSxtrheadlongpl{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 225

The behaviour of \GLSfmtlongpl when it occurs in a page header.

\Glsxtrheadlongpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

The behaviour of \Glsfmtlongpl when it occurs in a page header.

\glsxtrheadlongpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

The behaviour of \glsfmtlongpl when it occurs in a page header.

\GLSxtrheadname{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 227

Used to display the all caps entry's name field in the page header.

\Glsxtrheadname{⟨*entry-label*⟩} glossaries−extra v1.21+ §5.3.3; 227

Used to display the sentence case entry's name in the page header (converts to all caps if headuc attribute is true).

\glsxtrheadname{⟨*entry-label*⟩} glossaries−extra v1.21+ §5.3.3; 227

Used to display the entry's name in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSxtrheadplural{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 228

Used to display the all caps entry's `plural` field in the page header.

\Glsxtrheadplural{⟨*entry-label*⟩} §5.3.3; 228

Used to display the sentence case entry's `plural` field in the page header (converts to all caps if `headuc` attribute is `true`).

\glsxtrheadplural{⟨*entry-label*⟩} §5.3.3; 228

Used to display the entry's `plural` field in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSxtrheadshort{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 222

The behaviour of `\GLSfmtshort` when it occurs in a page header.

\Glsxtrheadshort{⟨*entry-label*⟩} §5.3.3; 222

The behaviour of `\Glsfmtshort` when it occurs in a page header.

\glsxtrheadshort{⟨*entry-label*⟩} §5.3.3; 221

The behaviour of `\glsfmtshort` when it occurs in a page header.

\GLSxtrheadshortpl{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 224

The behaviour of `\GLSfmtshortpl` when it occurs in a page header.

\Glsxtrheadshortpl{⟨*entry-label*⟩} §5.3.3; 224

The behaviour of `\Glsfmtshortpl` when it occurs in a page header.

\glsxtrheadshortpl{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 223

The behaviour of `\glsfmtshortpl` when it occurs in a page header.

\GLSxtrheadtext{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.3.3; 228

Used to display the all caps entry's `text` field in the page header.

\Glsxtrheadtext{⟨*entry-label*⟩} §5.3.3; 228

Used to display the sentence case entry's `text` field in the page header (converts to all caps if `headuc` attribute is `true`).

\glsxtrheadtext{⟨*entry-label*⟩} §5.3.3; 227

Used to display the entry's `text` field in the page header (converts to all caps if `headuc` attribute is `true`).

\GLSXTRhiername{⟨*entry-label*⟩} glossaries−extra v1.37+ §5.11; 301

Displays the entry's hierarchical name where each name is converted to uppercase.

\GLSxtrhiername{⟨*entry-label*⟩} glossaries−extra v1.37+ §5.11; 301

Displays the entry's hierarchical name where the first name is converted to uppercase.

\GlsXtrhiername{⟨*entry-label*⟩} glossaries−extra v1.37+ §5.11; 301

Displays the entry's hierarchical name where each element name has its first character converted to uppercase.

\Glsxtrhiername{⟨*entry-label*⟩} glossaries−extra v1.37+ §5.11; 301

Displays the entry's hierarchical name using sentence case.

\glsxtrhiername{⟨*entry-label*⟩} glossaries−extra v1.37+ §5.11; 300

Displays the entry's hierarchical name.

\glsxtrhiernamesep glossaries−extra v1.37+ §5.11; 301

Separator used by commands like `\glsxtrhiername`.

\glsxtrhyphenIIrules glossaries−extra−bib2gls v1.56+ 592

Expands to ordered hyphen character sort rules.

\glsxtrhyphenIrules glossaries−extra−bib2gls v1.56+ 592

Expands to equivalent hyphen character sort rules (excluding minus signs).

\glsxtrhyphenrules glossaries−extra−bib2gls v1.27+ 591

Combines the hyphen character sort rules `\glsxtrhyphenIrules` and the minus sort rules `\glsxtrminusrules`.

\glsxtrhyphensuffix *initial:* `\glsxtrabbrvpluralsuffix`
glossaries−extra v1.17+ 154

The plural suffix used by the "hyphen" abbreviation styles (such as short−hyphen−long−hyphen).

\glsxtridentifyglslike{⟨*label-prefix*⟩}{⟨*cs*⟩} glossaries−extra v1.37+ §5.7; 265

Used to inform `bib2gls` to include the given command when it searches for dependencies.

\glsxtrifallcaps{ ⟨*all caps*⟩ } { ⟨*not all caps*⟩ } glossaries−extra v1.49+ §5.5.4; 258

Shortcut for \glscapscase{ ⟨*not all caps*⟩ } { ⟨*not all caps*⟩ } { ⟨*all caps*⟩ }.

\glsxtrifcounttrigger{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } §6.1; 321

Does ⟨*true*⟩ if the entry's total use count at the end of the previous run exceeds the trigger value assigned to the entry's category, otherwise does ⟨*false*⟩.

\glsxtrifcustomdiscardperiod{ ⟨*true*⟩ } { ⟨*false*⟩ } *initial:* ⟨*false*⟩ glossaries−extra v1.23+ §5.5.4; 253

User hook to trigger a check for a following full stop. This should do ⟨*true*⟩ if there should be a check for a following full stop otherwise should do ⟨*false*⟩.

\glsxtrifemptyglossary{ ⟨*glossary-type*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } §8; 385

Does ⟨*true*⟩ if the glossary identified by ⟨*glossary-type*⟩ is empty, otherwise does ⟨*false*⟩. If the glossary doesn't exist, this does ⟨*true*⟩ and will either generate an error (undefaction=error) or a warning (undefaction=warn). This command considers ignored glossaries as existing.

\GlsXtrIfFieldCmpNum{ ⟨*field-label*⟩ } { ⟨*entry-label*⟩ } { ⟨*op*⟩ } { ⟨*number*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } *modifier:* * glossaries−extra v1.31+ §5.15; 313

Compares the (numeric) value of the field identified by its internal label ⟨*field-label*⟩ for the entry identified by ⟨*entry-label*⟩ with ⟨*number*⟩ where ⟨*op*⟩ is the comparison operator (=, < or >). The unstarred version adds implicit grouping. The starred version doesn't.

\GlsXtrIfFieldEqNum{ ⟨*field-label*⟩ } { ⟨*entry-label*⟩ } { ⟨*number*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } *modifier:* * glossaries−extra v1.31+ §5.15; 314

A shortcut that uses \GlsXtrIfFieldCmpNum with ⟨*op*⟩ set to =. The unstarred version adds implicit grouping. The starred version doesn't.

\GlsXtrIfFieldEqStr{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.21+

§5.15; 314

Tests if the entry given by ⟨*entry-label*⟩ has the field identified by its internal label ⟨*field-label*⟩ set to ⟨*value*⟩. This internally uses `\glsxtrifhasfield` and compares `\glscurrentfieldvalue` to ⟨*value*⟩ using etoolbox's `\ifdefstring`. The unstarred version adds implicit grouping. The starred version doesn't.

\GlsXtrIfFieldEqXpStr{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.31+

§5.15; 314

Like `\GlsXtrIfFieldEqStr` but first (protected) expands ⟨*value*⟩.

\GlsXtrIfFieldNonZero{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.31+

§5.15; 314

A shortcut that uses `\GlsXtrIfFieldCmpNum` to test if the (numeric) value of the field identified by its internal label ⟨*field-label*⟩ for the entry identified by ⟨*entry-label*⟩ is non-zero. An empty or undefined field is treated as 0. The unstarred version adds implicit grouping. The starred version doesn't. The value can be referenced within ⟨*true*⟩ (where it will be 0) or within ⟨*false*⟩ using `\glscurrentfieldvalue`.

\GlsXtrIfFieldUndef{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.23+

§5.15; 313

Expandable command that tests if the given field (identified by its internal label) is undefined for the entry given by ⟨*entry-label*⟩. Internally uses etoolbox's `\ifcsundef` command. Unlike `\glsxtrifhasfield` there is no grouping or starred version.

\GlsXtrIfFieldValueInCsvList{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*csv-list*⟩}{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.42+

§5.13; 312

Tests if the value stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ is contained in the comma-separated list ⟨*csv-list*⟩ using `\DTLifinlist` (provided by datatool−base, which is automatically loaded by the glossaries package). One level expansion is performed on ⟨*csv-list*⟩. This command uses `\glsxtrifhasfield` so the field value can be obtained with `\glscurrentfieldvalue`. The unstarred version adds implicit grouping. The starred version doesn't.

\glsxtrifhasfield{ ⟨*field-label*⟩ } { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
modifier: * glossaries−extra v1.19+

§5.15; 313

Tests if the field identified by its internal label ⟨*field-label*⟩ for the entry given by ⟨*entry-label*⟩ is defined and is not empty. This is like `\ifglshasfield` but doesn't produce a warning if the entry or field doesn't exist. This sets `\glscurrentfieldvalue` to the field value and does ⟨*true*⟩ if its defined and not empty, otherwise it does ⟨*false*⟩. The unstarred version adds implicit grouping to make nesting easier. The starred version doesn't (to make assignments easier).

\GlsXtrIfHasNonZeroChildCount{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
modifier: * glossaries−extra−bib2gls v1.47+

§11.6.4; 603

Tests if the value in the `childcount` field is non-zero (using `\GlsXtrIfFieldNonZero`). This requires the `save-child-count` resource option.

\glsxtrifheaduc{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.49+

§5.3.3; 221

If the category associated with the entry given by ⟨*entry-label*⟩ has the `headuc` attribute set to `true` this does ⟨*true*⟩ otherwise it does ⟨*false*⟩.

\glsxtrifhyphenstart{ ⟨*text*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.17+

§4.5.2; 166

If ⟨*text*⟩ starts with a hyphen this does ⟨*true*⟩ otherwise it does ⟨*false*⟩.

\glsxtrifindexing{ ⟨*true*⟩ } { ⟨*false*⟩ }

§5.8; 273

Tests whether or not the `noindex` has been set. Does ⟨*false*⟩ if `noindex=true` otherwise does ⟨*true*⟩.

\GlsXtrIfInGlossary{ ⟨*entry-label*⟩ } { ⟨*glossary-type*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.49+

§8; 384

Does ⟨*true*⟩ if the entry given by ⟨*entry-label*⟩ is in the internal list of the glossary identified by ⟨*glossary-type*⟩, otherwise it does ⟨*false*⟩. If the glossary doesn't exist, this does ⟨*false*⟩ and will either generate an error (`undefaction=error`) or a warning (`undefaction=warn`). This command considers ignored glossaries as existing.

\glsxtrifinlabelprefixlist { ⟨*label-prefix*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra−bib2gls v1.37+

§11.6.7; 613

Does ⟨*true*⟩ if ⟨*label-prefix*⟩ has been identified as a label prefix.

\glsxtrifinmark { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.07+

§5.3.3; 218

Does ⟨*true*⟩ if within `\markright`, `\markboth` or `\@starttoc` otherwise does ⟨*false*⟩.

\glsxtrifintoc { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.49+

§5.3.3; 218

Normally just expands to ⟨*false*⟩ but `\@starttoc` is redefined to temporarily set this macro to expand to ⟨*true*⟩ instead. Will always expand to ⟨*false*⟩ if `\glsxtrRevertTocMarks` or `\glsxtrRevertMarks` are used to revert `\@starttoc` to its former definition.

\glsxtrifkeydefined { ⟨*key*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.12+

§3.2; 36

Tests if the given ⟨*key*⟩ has been defined as a glossary entry key. Does ⟨*true*⟩ if the key has been defined, otherwise does false.

\glsxtriflabelinlist { ⟨*label*⟩ } { ⟨*label-list*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.21+

§8.4.3; 416

Does ⟨*true*⟩ if the given label is in the given comma-separated list of labels, otherwise does ⟨*false*⟩. The label and list are fully expanded.

\GlsXtrIfLinkCounterDef { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.26+

§6.2; 328

Expands to ⟨*true*⟩ if the link counter associated with the given entry has been defined, otherwise expands to ⟨*false*⟩.

\glsxtrifmulti { ⟨*multi-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+

§7.13; 379

Does ⟨*true*⟩ if a multi-entry has been defined with the label ⟨*multi-label*⟩ otherwise does ⟨*false*⟩.

\glsxtrifnextpunc{⟨*true*⟩}{⟨*false*⟩} glossaries−extra

§5.5.4; 254

Performs ⟨*true*⟩ if this command is followed by a recognised punctuation character otherwise does ⟨*false*⟩. The list of recognised punctuation marks is initialised to `.,:;?!` (full stop, comma, colon, semicolon, question mark, and exclamation mark). Additional punctuation characters can be added with `\glsxtraddpunctuationmark`.

\glsxtrifperiod{⟨*true*⟩}{⟨*false*⟩}⟨*token*⟩ glossaries−extra

§5.5.4; 253

Does ⟨*true*⟩ if ⟨*token*⟩ is a full stop, otherwise does ⟨*false*⟩.

\glsxtrifrecordtrigger{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
glossaries−extra v1.21+

§11.5; 570

Does ⟨*true*⟩ if the entry's total record count (obtained with `\glsxtrrecordtriggervalue`) exceeds the value supplied by the `recordcount` attribute, otherwise does ⟨*false*⟩.

\GlsXtrIfUnusedOrUndefined{⟨*entry-label*⟩}{⟨*true*⟩}{⟨*false*⟩}
glossaries−extra v1.34+

§5.10; 292

Does ⟨*true*⟩ if the entry hasn't been defined or hasn't been marked as used, otherwise does ⟨*true*⟩. Note that this command will generate an error or warning (according to `undefaction`) if the entry hasn't been defined, but will still do ⟨*true*⟩.

\GlsXtrIfValueInFieldCsvList{⟨*entry-label*⟩}{⟨*field-label*⟩}
{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.47+

§5.13; 311

Tests if the given value (⟨*value*⟩) is contained in the comma-separated list stored in the given field (identified by its internal label) for the entry identified by ⟨*entry-label*⟩ using `\DTLifinlist` (provided by datatool−base, which is automatically loaded by the glossaries package). No expansion is performed on ⟨*value*⟩. This command uses `\glsxtrifhasfield` so the complete list can be obtained with `\glscurrentfieldvalue`. The unstarred version adds implicit grouping. The starred version doesn't.

\glsxtrifwasfirstuse{⟨*true*⟩}{⟨*false*⟩}

§5.10; 291

Initialised by the `\gls`-like and `\glstext`-like commands, this expands to ⟨*true*⟩ if the

calling command was considered the first use, otherwise it expands to ⟨*false*⟩. This command may be used within the post-link hook (where it's too late to test the first use flag with `\ifglsused`).

`\glsxtrifwasglslike`{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §5.5.4; 257

Initialised by the `\gls`-like and `\glstext`-like commands, this expands to ⟨*true*⟩ if the calling command was a `\gls`-like command, otherwise it expands to ⟨*false*⟩. This command may be used within the post-link hook.

`\glsxtrifwasglslikeandfirstuse`{⟨*true*⟩}{⟨*false*⟩}
glossaries−extra v1.49+ §5.5.4; 258

A shortcut that nests `\glsxtrifwasglslike` and `\glsxtrifwasfirstuse`. This does ⟨*true*⟩ if the calling command was both a `\gls`-like command and was considered the first use.

`\glsxtrifwassubsequentorshort`{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §5.5.4; 258

Expands to ⟨*true*⟩ if the calling command was a `\gls`-like command and was the subsequent use or if `\glsxtrcurrentfield` was set to `short`.

`\glsxtrifwassubsequentuse`{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.49+ §5.5.4; 258

A shortcut that nests `\glsxtrifwasglslike` and `\glsxtrifwasfirstuse`. This does ⟨*true*⟩ if the calling command was a `\gls`-like command but was not considered the first use.

`\GlsXtrIfXpFieldEqXpStr`{⟨*field-label*⟩}{⟨*entry-label*⟩}{⟨*value*⟩}
{⟨*true*⟩}{⟨*false*⟩} *modifier:* * glossaries−extra v1.31+ §5.15; 314

Like `\GlsXtrIfFieldEqStr` but first (protected) expands both the field value and the supplied ⟨*value*⟩.

`\glsxtrIgnorableRules` glossaries−extra−bib2gls v1.49+ 595

A shortcut that expands to the control rules, space rules and non-printable rules.

\glsxtrinclinkcounter{⟨*counter*⟩} glossaries−extra v1.26+ §6.2; 327

Increments the link counter with `\stepcounter`.

\glsxtrindexaliased glossaries−extra v1.12+ §5.9.3; 287

Index the current entry's alias. May only be used within the definition of `\glsxtrsetaliasnoindex`.

\GlsXtrIndexCounterLink{⟨*text*⟩}{⟨*entry-label*⟩}
glossaries−extra−bib2gls v1.29+ §11.6.8; 622

Creates a hyperlink (if supported) to the target obtained from `indexcounter`, if the field has been defined with the given hyperlink text (otherwise just does ⟨*text*⟩).

\glsxtrindexseealso{⟨*entry-label*⟩}{⟨*xr-list*⟩} glossaries−extra v1.16+ §5.9.3; 287

Indexes the entry identified by ⟨*entry-label*⟩ as a "see also" cross-reference to the entries identified in the comma-separated list ⟨*xr-list*⟩. The cross-reference list is prefixed with `\seealsoname`.

\glsxtrinithyperoutside glossaries−extra v1.21+ §5.1.1; 193

Hook that initialises the `hyperoutside` setting.

\glsxtrinitwrgloss glossaries−extra v1.14+ §5.1.1; 193

Hook that initialises the `wrgloss` setting.

\glsxtrinitwrglossbeforefalse glossaries−extra v1.14+

Corresponds to `wrgloss=after`.

\glsxtrinitwrglossbeforetrue glossaries−extra v1.14+

Corresponds to `wrgloss=before`.

\GLSxtrinlinefullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 180

Used by \GLSxtrfull to display the all caps inline full form form (defined by the abbreviation style).

\Glsxtrinlinefullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 180

Used by \Glsxtrfull to display the sentence case inline full form form (defined by the abbreviation style).

\glsxtrinlinefullformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 180

Used by \glsxtrfull to display the inline full form form (defined by the abbreviation style).

\GLSxtrinlinefullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.49+ 180

Used by \GLSxtrfullpl to display the plural all caps inline full form form (defined by the abbreviation style).

\Glsxtrinlinefullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 180

Used by \Glsxtrfullpl to display the plural sentence case inline full form form (defined by the abbreviation style).

\glsxtrinlinefullplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } 180

Used by \glsxtrfullpl to display the plural inline full form form (defined by the abbreviation style).

\glsxtrinsertinsidefalse glossaries−extra v1.02 138

Sets the \ifglsxtrinsertinside conditional to false.

\glsxtrinsertinsidetrue glossaries−extra v1.02 138

Sets the `\ifglsxtrinsertinside` to true.

\GlsXtrInternalLocationHyperlink{ ⟨*counter*⟩ } { ⟨*prefix*⟩ } { ⟨*location*⟩ } glossaries−extra v1.29+

Used by `\glsxtrlocationhyperlink` to create an internal hyperlink to the given location (advanced command, see documented code for use).

\glsxtrLatinA glossaries−extra−bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin A (includes 0x00AA and 0x2090).

\glsxtrLatinAA glossaries−extra−bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin å.

\glsxtrLatinAELigature glossaries−extra−bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin ae-ligature.

\glsxtrLatinE glossaries−extra−bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin E (includes 0x2091).

\glsxtrLatinEszettSs glossaries−extra−bib2gls v1.27+ 599

(Sort rule) expands to rule for ßand fs.

\glsxtrLatinEszettSz glossaries−extra−bib2gls v1.27+ 599

(Sort rule) expands to rule for ßand fz.

\glsxtrLatinEth glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin eth.

\glsxtrLatinH glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin H (includes 0x2095).

\glsxtrLatinI glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin I (includes 0x2071).

\glsxtrLatinInsularG glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin insular G and g, G.

\glsxtrLatinK glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin K (includes 0x2096).

\glsxtrLatinL glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of Latin L (includes 0x2097).

\glsxtrLatinLslash glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin ł.

\glsxtrLatinM glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin M (includes 0x2098).

\glsxtrLatinN glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin N (includes 0x2099).

\glsxtrLatinO glossaries-extra-bib2gls v1.27+ 598

(Sort rule) expands to the variations of Latin O (includes 0x00BA and 0x2092).

\glsxtrLatinOELigature glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin oe-ligature.

\glsxtrLatinOslash glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin ø.

\glsxtrLatinP glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin P (includes 0x209A).

\glsxtrLatinS glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin S (includes 0x209B).

\glsxtrLatinSchwa glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin schwa.

\glsxtrLatinT glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin T (includes 0x209C).

\glsxtrLatinThorn glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin thorn.

\glsxtrLatinWynn glossaries-extra-bib2gls v1.27+ 600

(Sort rule) expands to the variations of Latin wynn.

\glsxtrLatinX glossaries-extra-bib2gls v1.27+ 599

(Sort rule) expands to the variations of Latin X (includes 0x2093).

\GlsXtrLetField{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*cs*⟩ } glossaries-extra v1.12+ §3.5; 41

Like `\GlsXtrSetField` but internally uses (etoolbox's) `\cslet` instead of `\csdef`.

\GlsXtrLetFieldToField{ ⟨*entry1-label*⟩ } { ⟨*field1-label*⟩ } { ⟨*entry2-label*⟩ } { ⟨*field2-label*⟩ } glossaries-extra v1.12+ §3.5; 41

Assigns the field identified by its internal label ⟨*field1-label*⟩ for the entry identified by ⟨*entry1-label*⟩ to the value of the field identified by ⟨*field2-label*⟩ for the entry identified by ⟨*entry2-label*⟩.

\GlsXtrLinkCounterName{ ⟨*entry-label*⟩ } glossaries-extra v1.26+ §6.2; 328

Expands to the name of the link counter associated with the given entry (no check for existence).

\GlsXtrLinkCounterValue{ ⟨*entry-label*⟩ } glossaries-extra v1.26+ §6.2; 327

Expands to the internal link count register associated with the given register or 0 if it hasn't been defined.

\GlsXtrLoadResources [⟨*options*⟩] glossaries-extra v1.11+ §11; 541

For use with `bib2gls`, this both sets up the options for the resource set (which `bib2gls` can detect from the `aux` file) and inputs the file ⟨*basename*⟩`.glstex` created by `bib2gls` (if

it exists), where the ⟨*basename*⟩ is obtained from `\jobname` and `\glsxtrresourcecount`.

`\glsxtrlocalsetgrouptitle`{⟨*group-label*⟩}{⟨*group-title*⟩}
glossaries−extra v1.24+

§8.6.4; 443

Locally assigns the given title ⟨*group-title*⟩ to the group identified by ⟨*group-label*⟩.

`\glsxtrlocationanchor` glossaries−extra−bib2gls v1.49+

§11.6.6; 606

Defined by `\glsxtrdisplaylocnameref` to expand to the anchor constructed from ⟨*counter*⟩ and ⟨*hcounter*⟩, which corresponds to the record counter.

`\GlsXtrLocationField` *initial:* `location` glossaries−extra v1.37+

§8.4.2; 410

Expands to the internal field label used to obtain the formatted location list for the "unsrt" family of commands.

`\glsxtrlocationhyperlink`{⟨*counter*⟩}{⟨*prefix*⟩}{⟨*location*⟩}
glossaries−extra v1.14+

Used to create a hyperlink to either an external or an internal location, depending on whether or not `\glsxtrsupplocationurl` is defined and not empty (advanced command, see documented code for use).

`\GlsXtrLocationRecordCount`{⟨*entry-label*⟩}{⟨*counter*⟩}{⟨*location*⟩}
glossaries−extra v1.21+

§11.5; 570

Expands to the entry's record count for the given counter and location (stored in the `recordcount.`⟨*counter*⟩`.`⟨*location*⟩ field) or to 0 if not set.

`\glsxtrlocrangefmt` glossaries−extra v1.12+

§8.6.3; 441

Expands to the range format (set by `\glsxtrdisplaystartloc`).

\GLSxtrlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrlong but converts the link text to all caps.

\Glsxtrlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrlong but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glsxtrlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 53

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the long value, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\GLSxtrlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 181

As \glsxtrlongformat but sentence case.

\Glsxtrlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 181

As \glsxtrlongformat but sentence case.

\glsxtrlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 180

Encapsulates the long field for the given entry with ⟨*fmt-cs*⟩. The ⟨*insert*⟩ argument is the insertion material supplied in the final optional argument of the \gls-like or \glstext-like commands. The \ifglsxtrinsertinside, inner formatting, and accessibility settings are supported.

\GLSxtrlongformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 181

As \glsxtrlongformatgrp but all caps.

\Glsxtrlongformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 181

As \glsxtrlongformatgrp but sentence case.

\glsxtrlongformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 181

As \glsxtrlongformat but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\glsxtrlonghyphen{⟨*entry-label*⟩}{⟨*long*⟩}{⟨*insert*⟩} glossaries−extra v1.17+ 156

Formats the long form according to the long−hyphen−postshort−hyphen style.

\GLSxtrlonghyphennoshort{⟨*entry-label*⟩}{⟨*long*⟩}{⟨*insert*⟩}
glossaries−extra v1.49+ 155

As \glsxtrlonghyphennoshort but converts ⟨*insert*⟩ to all caps The ⟨*long*⟩ argument should be supplied as all caps.

\glsxtrlonghyphennoshort{⟨*entry-label*⟩}{⟨*long*⟩}{⟨*insert*⟩}
glossaries−extra v1.17+ 155

Formats the long form according to the long−hyphen−noshort−desc−noreg style.

\glsxtrlonghyphennoshortdescsort glossaries−extra v1.49+ 155

Expands to the sort value for the long−hyphen−noshort−desc−noreg styles.

\glsxtrlonghyphennoshortsort glossaries−extra v1.49+ 154

Expands to the sort value for the long−hyphen−noshort−noreg styles.

\GLSxtrlonghyphenshort { ⟨*entry-label*⟩ } { ⟨*long*⟩ } { ⟨*short*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.49+ 155

As \glsxtrlonghyphenshort but converts ⟨*insert*⟩ to all caps. The ⟨*long*⟩ and ⟨*short*⟩ arguments should be supplied as all caps.

\glsxtrlonghyphenshort { ⟨*entry-label*⟩ } { ⟨*long*⟩ } { ⟨*short*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.17+ 155

Formats the long and short form according to the long−hyphen−short−hyphen style.

\glsxtrlonghyphenshortsort glossaries−extra v1.49+ 154

Expands to the sort value for the long−hyphen−short−hyphen styles.

\glsxtrlongnoshortdescname glossaries−extra v1.25+ 153

Expands to the name value for styles like long−noshort−desc.

\glsxtrlongnoshortname glossaries−extra v1.25+ 153

Expands to the name value for styles like long−noshort.

\GLSxtrlongpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrlongpl but converts the link text to all caps.

\Glsxtrlongpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 54

As \glsxtrlongpl but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glsxtrlongpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

§4.3; 54

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `long-plural` value, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options.

\GLSxtrlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

181

As `\glsxtrlongplformat` but all caps.

\Glsxtrlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

181

As `\glsxtrlongplformat` but sentence case.

\glsxtrlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

181

As `\glsxtrlongformat` but for the `longplural` field.

\GLSxtrlongplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

182

As `\glsxtrlongplformatgrp` but all caps.

\Glsxtrlongplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

182

As `\glsxtrlongplformatgrp` but sentence case.

\glsxtrlongplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 181

As \glsxtrlongplformat but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\glsxtrlongshortdescname glossaries−extra v1.17+ 140

Expands to the name value for long−short−desc styles.

\glsxtrlongshortdescsort glossaries−extra v1.04+ 140

Expands to the sort value for long−short−desc styles.

\GLSxtrlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ }
{ ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

As \glsxtrlongshortformat but all caps.

\Glsxtrlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ }
{ ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

As \glsxtrlongshortformat but sentence case.

\glsxtrlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ }
{ ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 183

Formats the long form with \glsxtrlongformat and the short form in parentheses with \glsxtrshortformat.

\glsxtrlongshortname glossaries−extra v1.25+ 140

Expands to the name value for long−short styles.

\GLSxtrlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

As \glsxtrlongshortplformat but all caps.

\Glsxtrlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

As \glsxtrlongshortplformat but sentence case.

\glsxtrlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

As \glsxtrlongshortformat but for the plurals.

\glsxtrlongshortscuserdescname glossaries−extra v1.48+ 145

Expands to the value for the name key for styles like long−postshort−sc−user−desc.

\glsxtrlongshortscusername glossaries−extra v1.48+ 145

Expands to the value for the name key for styles like long−postshort−sc−user.

\glsxtrlongshortuserdescname glossaries−extra v1.25+ 144

Expands to the value for the name key for styles like long−short−user−desc.

\glsxtrmarkhook §5.3.3; 230

Hook that's performed at the start of \markright, \markboth and \@starttoc to redefine commands that need to change when they occur within page headers or contents. This must be counteracted with \glsxtrrestoremarkhook afterwards.

\glsxtrMathGreekIIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the second set of math Greek sort rules.

\glsxtrMathGreekIrules glossaries-extra-bib2gls v1.27+ 600

Expands to the first set of math Greek sort rules.

\glsxtrMathItalicAlpha glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek alpha.

\glsxtrMathItalicBeta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek beta.

\glsxtrMathItalicChi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek chi.

\glsxtrMathItalicDelta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek delta.

\glsxtrMathItalicDigamma glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek digamma.

\glsxtrMathItalicEpsilon glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek epsilon.

\glsxtrMathItalicEta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek eta.

\glsxtrMathItalicGamma glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek gamma.

\glsxtrMathItalicGreekIIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the second set of math italic Greek sort rules.

\glsxtrMathItalicGreekIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the first set of math italic Greek sort rules.

\glsxtrMathItalicIota glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek iota.

\glsxtrMathItalicKappa glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek kappa.

\glsxtrMathItalicLambda glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek lambda.

\glsxtrMathItalicLowerGreekIIrules glossaries-extra-bib2gls v1.27+ 602

Expands to the second set of math italic lowercase Greek sort rules.

\glsxtrMathItalicLowerGreekIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the first set of math italic lowercase Greek sort rules.

\glsxtrMathItalicMu glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek mu.

\glsxtrMathItalicNabla glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the Unicode codepoint for nabla.

\glsxtrMathItalicNu glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek nu.

\glsxtrMathItalicOmega glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek omega.

\glsxtrMathItalicOmicron glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek omicron.

\glsxtrMathItalicPartial glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the Unicode codepoint for math italic partial differential.

\glsxtrMathItalicPhi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek phi.

\glsxtrMathItalicPi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek pi.

\glsxtrMathItalicPsi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek psi.

\glsxtrMathItalicRho glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek rho.

\glsxtrMathItalicSigma glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek sigma.

\glsxtrMathItalicTau glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek tau.

\glsxtrMathItalicTheta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek theta.

\glsxtrMathItalicUpperGreekIIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the second set of math italic uppercase Greek sort rules.

\glsxtrMathItalicUpperGreekIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the first set of math italic uppercase Greek sort rules.

\glsxtrMathItalicUpsilon glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek upsilon.

\glsxtrMathItalicXi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek xi.

\glsxtrMathItalicZeta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math italic Greek zeta.

\glsxtrMathUpGreekIIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the second set of math upright Greek sort rules.

\glsxtrMathUpGreekIrules glossaries-extra-bib2gls v1.27+ 601

Expands to the first set of math upright Greek sort rules.

\glsxtrMFUsave glossaries-extra v1.49+ §11; 543

Used on the first instance of `\GlsXtrLoadResources`, this will add `\MFUsave` to the begin document hook and then disable itself. This is provided to help `bib2gls` pick up any of mfirstuc's exclusions, blockers and mappings to assist with its sentence case function.

\GlsXtrMglsOrGls { ⟨*mgls cs*⟩ } { ⟨*gls cs*⟩ } ⟨*modifier*⟩ [⟨*options*⟩] { ⟨*label*⟩ } [⟨*insert*⟩] glossaries-extra v1.48+ §7; 335

If ⟨*label*⟩ matches a defined multi-entry, this will do ⟨*mgls cls*⟩ otherwise it will do ⟨*gls cs*⟩. The ⟨*modifier*⟩ (* or + or the token identified with `\GlsXtrSetAltModifier`) may be omitted.

\glsxtrmglsWarnAllSkipped{ ⟨*message*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.48+

§7.9.5; 367

Issues the given warning message with `\GlossariesExtraWarning` and does ⟨*fmt-cs*⟩ { ⟨*insert*⟩ } (this warning is used if all elements of a multi-entry set are skipped).

\glsxtrminusrules glossaries−extra−bib2gls v1.56+

592

Expands to minus character sort rules (excluding the ASCII hyphen/minus character).

\GLSxtrmultientryadjustedname{ ⟨*sublist1*⟩ } { ⟨*name*⟩ } { ⟨*sublist2*⟩ } { ⟨*multi-label*⟩ } glossaries−extra−bib2gls v1.48+

§7.14; 381

As `\glsxtrmultientryadjustedname` but all caps.

\GlsXtrmultientryadjustedname{ ⟨*sublist1*⟩ } { ⟨*name*⟩ } { ⟨*sublist2*⟩ } { ⟨*multi-label*⟩ } glossaries−extra−bib2gls v1.48+

§7.14; 381

As `\glsxtrmultientryadjustedname` but title case.

\Glsxtrmultientryadjustedname{ ⟨*sublist1*⟩ } { ⟨*name*⟩ } { ⟨*sublist2*⟩ } { ⟨*multi-label*⟩ } glossaries−extra−bib2gls v1.48+

§7.14; 381

As `\glsxtrmultientryadjustedname` but sentence case.

\glsxtrmultientryadjustedname{ ⟨*sublist1*⟩ } { ⟨*name*⟩ } { ⟨*sublist2*⟩ } { ⟨*multi-label*⟩ } glossaries−extra−bib2gls v1.48+

§7.14; 380

Used by `compound-adjust-name`.

\GLSxtrmultientryadjustednamefmt{ ⟨*text*⟩ }
glossaries−extra−bib2gls v1.48+

§7.14; 382

Used by `\GLSxtrmultientryadjustedname` to encapsulate the main entry name.

\GlsXtrmultientryadjustednamefmt{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 382

Used by \GlsXtrmultientryadjustedname to encapsulate the main entry name.

\Glsxtrmultientryadjustednamefmt{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 381

Used by \Glsxtrmultientryadjustedname to encapsulate the main entry name if the first sublist is empty.

\glsxtrmultientryadjustednamefmt{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 381

Used by \glsxtrmultientryadjustedname to encapsulate the main entry name.

\GLSxtrmultientryadjustednameother{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 382

Used by \GLSxtrmultientryadjustedname to encapsulate the other (not main) entries.

\GlsXtrmultientryadjustednameother{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 382

Used by \GlsXtrmultientryadjustedname to encapsulate the other (not main) entries.

\Glsxtrmultientryadjustednameother{⟨*text*⟩}
glossaries-extra-bib2gls v1.48+

§7.14; 382

Used by \Glsxtrmultientryadjustedname to encapsulate the other (not main) entries.

\glsxtrmultientryadjustednameother{ ⟨*text*⟩ }
glossaries−extra−bib2gls v1.48+ §7.14; 382

Used by \glsxtrmultientryadjustedname to encapsulate the other (not main) entries.

\glsxtrmultientryadjustednamepostsep{ ⟨*pre-label*⟩ } { ⟨*post-label*⟩ }
glossaries−extra−bib2gls v1.48+ §7.14; 381

Separator used by \glsxtrmultientryadjustedname between the main element and the first element of the second sublist.

\glsxtrmultientryadjustednamepresep{ ⟨*pre-label*⟩ } { ⟨*post-label*⟩ }
glossaries−extra−bib2gls v1.48+ §7.14; 381

Separator used by \glsxtrmultientryadjustedname between the last element of the first sublist and the main element.

\glsxtrmultientryadjustednamesep{ ⟨*pre-label*⟩ } { ⟨*post-label*⟩ }
glossaries−extra−bib2gls v1.48+ §7.14; 381

Separator used by \glsxtrmultientryadjustedname.

\glsxtrmultilastotherindex{ ⟨*multi-label*⟩ }
glossaries−extra v1.48+ §7.13; 379

Expands to the index of the final non-main element in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined.

\glsxtrmultilist{ ⟨*multi-label*⟩ }
glossaries−extra v1.48+ §7.13; 379

Expands to the list of element labels for the multi-entry identified by ⟨*multi-label*⟩ or nothing if not defined.

\glsxtrmultimain{ ⟨*multi-label*⟩ }
glossaries−extra v1.48+ §7.13; 379

Expands to the main label for the multi-entry identified by ⟨*multi-label*⟩ or nothing if not defined.

\glsxtrmultimainindex{ ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.13; 379

Expands to the index of the main element in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined.

\glsxtrmultisupplocation{ ⟨*src*⟩ } { ⟨*location*⟩ } { ⟨*format*⟩ } *modifier:* * glossaries−extra−bib2gls v1.36+ §11.6.5; 604

Used by `\glsxtrdisplaysupploc` to format the location.

\glsxtrmultitotalelements{ ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.13; 379

Expands to the total number of elements in the given multi-entry or nothing if ⟨*multi-label*⟩ hasn't been defined.

\glsxtrnameloclink{ ⟨*prefix*⟩ } { ⟨*counter*⟩ } { ⟨*format*⟩ } { ⟨*location*⟩ } { ⟨*text*⟩ } { ⟨*file*⟩ } glossaries−extra−bib2gls v1.37+ §11.6.6; 609

Create an external location hyperlink using the prefix and counter.

\glsxtrnamereflink{ ⟨*format*⟩ } { ⟨*title*⟩ } { ⟨*target*⟩ } { ⟨*file*⟩ } glossaries−extra−bib2gls v1.37+

Used by `\glsxtrdisplaylocnameref` to create a location hyperlink.

\glsxtrnewabbrevpresetkeyhook{ ⟨*options*⟩ } { ⟨*label*⟩ } { ⟨*short*⟩ } §4.1.5; 46

Hook provided to adjust initialisation within `\newabbreviation`.

\glsxtrnewgls [⟨*default-options*⟩] { ⟨*prefix*⟩ } { ⟨*cs*⟩ } glossaries−extra v1.21+ §5.7; 263

Defines the command ⟨*cs*⟩[⟨*options*⟩]{⟨*entry-label*⟩} to behave like `\gls` [⟨*default-options*⟩**,** ⟨*options*⟩] { ⟨*prefix*⟩⟨*entry-label*⟩ }.

\glsxtrnewglsdisp [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*cs*⟩} glossaries-extra v1.49+ §5.7; 264

Defines the command ⟨*cs*⟩ [⟨*options*⟩] {⟨*label*⟩} {⟨*text*⟩} to behave like \glsdisp [⟨*default-options*⟩,⟨*options*⟩] {⟨*prefix*⟩⟨*label*⟩} {⟨*text*⟩}.

\glsxtrnewGLSlike [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*GLS-like cs*⟩} {⟨*GLSpl-like cs*⟩} glossaries-extra v1.21+ §5.7; 264

Like \glsxtrnewgls but provides all caps commands.

\glsxtrnewglslike [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*gls-like cs*⟩} {⟨*glspl-like cs*⟩} {⟨*Gls-like cs*⟩} {⟨*Glspl-like cs*⟩} glossaries-extra v1.21+ §5.7; 264

Like \glsxtrnewgls but provides plural and sentence case commands as well.

\glsxtrnewglslink [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*cs*⟩} glossaries-extra v1.49+ §5.7; 264

Defines the command ⟨*cs*⟩ [⟨*options*⟩] {⟨*label*⟩} {⟨*text*⟩} to behave like \glslink [⟨*default-options*⟩,⟨*options*⟩] {⟨*prefix*⟩⟨*label*⟩} {⟨*text*⟩}.

\glsxtrnewnumber [⟨*key=value list*⟩] {⟨*entry-label*⟩} {⟨*num*⟩} (requires \usepackage[numbers]{glossaries-extra}) §2.1; 12

Defines a new glossary entry with the given label, type set to numbers, the category set to number, the name set to ⟨*num*⟩ and the sort set to ⟨*entry-label*⟩. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.

\glsxtrnewrgls [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*cs*⟩} glossaries-extra v1.21+ §5.7; 265

Like \glsxtrnewgls but uses \rgls.

\glsxtrnewrGLSlike [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*rGLS-like cs*⟩} {⟨*rGLSpl-like cs*⟩} glossaries-extra v1.21+ §5.7; 265

Like \glsxtrnewrgls but provides all caps commands.

\glsxtrnewrglslike [⟨*default-options*⟩] {⟨*prefix*⟩} {⟨*rgls-like cs*⟩} {⟨*rglspl-like cs*⟩} {⟨*rGls-like cs*⟩} {⟨*rGlspl-like cs*⟩}
glossaries−extra v1.21+

§5.7; 265

Like \glsxtrnewrgls but provides plural and sentence case commands as well.

\glsxtrnewsymbol [⟨*key=value list*⟩] {⟨*entry-label*⟩} {⟨*sym*⟩} (requires \usepackage[symbols]{glossaries-extra})

§2.1; 11

Defines a new glossary entry with the given label, type set to symbols, the category set to symbol, the name set to ⟨*sym*⟩ and the sort set to ⟨*entry-label*⟩. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.

\glsxtrNoGlossaryWarning {⟨*glossary-type*⟩}

Issues a warning with \GlossariesExtraWarning indicating that the given glossary is missing.

\GlsXtrNoGlsWarningAutoMake {⟨*glossary-label*⟩}

§15; 645

Advisory message when automake has been used.

\GlsXtrNoGlsWarningBuildInfo

§15; 645

Build advice.

\GlsXtrNoGlsWarningCheckFile {⟨*file*⟩}

§15; 645

Advisory message to check the file contents.

\GlsXtrNoGlsWarningEmptyMain

§15; 644

Produces the boilerplate text if the probably empty glossary is the main one.

\GlsXtrNoGlsWarningEmptyNotMain{ ⟨*glossary-label*⟩ } §15; 644

Produces the boilerplate text if the probably empty glossary is not the `main` one.

\GlsXtrNoGlsWarningEmptyStart §15; 644

Produces the boilerplate text if a glossary is probably empty.

\GlsXtrNoGlsWarningHead{ ⟨*glossary-label*⟩ } { ⟨*file*⟩ } §15; 644

Produces the header boilerplate text if a glossary file is missing.

\GlsXtrNoGlsWarningMisMatch §15; 645

Advisory message on mis-matching `\makenoidxglossaries`.

\GlsXtrNoGlsWarningNoOut{ ⟨*file*⟩ } §15; 645

Advisory if no output file was created.

\GlsXtrNoGlsWarningTail §15; 645

Final paragraph of missing glossary boilerplate text.

\glsxtrnoidxgroups glossaries−extra v1.49+ §8.4; 395

Makes the group titling mechanism used with the "unsrt" family of commands use the same method as for `\printnoidxglossary`. This command can't be used with `\makeglossaries` or with `record`.

\glsxtrnonprintablerules glossaries−extra−bib2gls v1.27+ 590

Expands to non-printable character sort rules.

\glsxtrnopostpunc glossaries−extra v1.22+ §8.6.2; 439

When placed at the end of the description, this switches off the post-description punctuation (inserted automatically via options such as postdot) but doesn't suppress the post-description hook. Does nothing outside of the glossary.

\glsxtronlydescname glossaries−extra v1.17+ 161

Expands to the name value for styles like long−only−short−only−desc.

\glsxtronlydescsort glossaries−extra v1.17+ 161

Expands to the name value for styles like long−only−short−only−desc.

\glsxtronlyname glossaries−extra v1.25+ 161

Expands to the name value for styles like long−only−short−only.

\glsxtronlysuffix *initial:* \glsxtrabbrvpluralsuffix
glossaries−extra v1.17+ 160

The plural suffix used by the "only" abbreviation styles (such as long−only−short−only).

\glsxtrorgkeylist glossaries−extra v1.49+ §4.5.3.1; 169

Expands to the original option list as it was supplied to \newabbreviation.

\glsxtrorgshort glossaries−extra v1.17+ §4.5.3.1; 169

Expands to the original short form as it was supplied to \newabbreviation.

\glsxtrorglong glossaries−extra v1.17+ §4.5.3.1; 170

Expands to the original long form as it was supplied to \newabbreviation.

\GLSxtrp{⟨*field*⟩}{⟨*entry-label*⟩} glossaries−extra v1.07+ §5.4; 235

As \glsxtrp but converts to uppercase (but not in the PDF bookmark).

\Glsxtrp{⟨*field*⟩}{⟨*entry-label*⟩} glossaries−extra v1.07+ §5.4; 235

As \glsxtrp but converts the first letter to uppercase (but not in the PDF bookmark).

\glsxtrp{⟨*field*⟩}{⟨*entry-label*⟩} glossaries−extra v1.07+ §5.4; 234

For use in headings and captions (instead of the \gls-like or \glstext-like commands). This command is designed to expand to the field value if used in a PDF bookmark and can also expand to a more appropriate command if it ends up in the page header. Note that there's no optional argument. Options should be set beforehand using \glsxtrsetpopts, which is done automatically in the glossary with \glossxtrsetpopts.

\glsxtrpageref{⟨*entry-label*⟩} glossaries−extra v1.11+ §8.1; 385

As \glsrefentry but uses \pageref instead of \ref. As with \glsrefentry, this will use \gls instead if the corresponding entry counter is disabled.

\glsxtrparen{⟨*text*⟩} glossaries−extra v1.17+ 138

Used to encapsulate ⟨*text*⟩ in parentheses.

\Glsxtrpdfentryfmt{⟨*entry-label*⟩}{⟨*text*⟩} glossaries−extra v1.49+ §5.12.2; 307

Does \MFUsentencecase{⟨*text*⟩}.

\glsxtrpdfentryfmt{⟨*entry-label*⟩}{⟨*text*⟩} glossaries−extra v1.42+ §5.12.2; 304

Just does ⟨*text*⟩.

\glsxtrpInit { ⟨*cs-name*⟩ } { ⟨*entry-label*⟩ } glossaries−extra v1.51+ §5.4; 234

Hook implemented at the start of `\glsxtrp` (and case-changing variants) inside the added scoping. By default this disables the post-link hook and ignores its arguments.

\Glsxtrpl [⟨*gls-options*⟩] [⟨*dfn-options*⟩] { ⟨*entry-label*⟩ } §13; 629

As `\glsxtrpl` but applies sentence case.

\glsxtrpl [⟨*gls-options*⟩] [⟨*dfn-options*⟩] { ⟨*entry-label*⟩ } §13; 629

As `\glsxtr` but shows the plural form.

\glsxtrpostabbrvfootnote { ⟨*entry-label*⟩ } { ⟨*fmt-code*⟩ }
glossaries−extra v1.49+ §4.5.2; 165

Command used in the post-link hook for styles like short−postfootnote.

\glsxtrpostdescabbreviation *initial: empty* §8.6.2; 438

The default post-description hook for the `abbreviation` category.

\glsxtrpostdescacronym *initial: empty* §8.6.2; 438

The default post-description hook for the `acronym` category.

\glsxtrpostdesc⟨*category*⟩ glossaries−extra §8.6.2; 438

The post-description hook associated with the category identified by the label ⟨*category*⟩.

\glsxtrpostdescgeneral *initial: empty* §8.6.2; 438

The default post-description hook for the `general` category.

\glsxtrpostdescindex *initial: empty* (requires \usepackage[index]{glossaries-extra})

§2.1; 13

The default post-description hook for the index category.

\glsxtrpostdescnumber *initial: empty* (requires \usepackage[numbers]{glossaries-extra})

§2.1; 12

The default post-description hook for the number category.

\glsxtrpostdescription

§8.6.2; 438

An additional hook used within \glspostdescription that implements the category post-description hook.

\glsxtrpostdescsymbol *initial: empty* (requires \usepackage[symbols]{glossaries-extra})

§2.1; 11

The default post-description hook for the symbol category.

\glsxtrpostdescterm *initial: empty*

§8.6.2; 438

The default post-description hook for the term category (which isn't used by glossaries−extra).

\glsxtrpostfootnotelongformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+

152

Used in the footnote text to format the long form for styles like short−postfootnote.

\GLSxtrposthyphenlong{⟨*entry-label*⟩}{⟨*insert*⟩} glossaries−extra v1.49+

159

As \glsxtrposthyphenlong but all caps.

\glsxtrposthyphenlong { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 159

Used within the post-link hook to format the long form according to the short−hyphen−postlong −hyphen style on first use.

\GLSxtrposthyphenlongpl { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 159

As \glsxtrposthyphenlongpl but all caps.

\glsxtrposthyphenlongpl { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 159

As \glsxtrposthyphenlong but shows the plural.

\GLSxtrposthyphenshort { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 157

As \glsxtrposthyphenshort but all caps.

\glsxtrposthyphenshort { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 157

Used within the post-link hook to format the short form according to the long−hyphen−postshort −hyphen style on first use.

\GLSxtrposthyphenshortpl { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 157

As \glsxtrposthyphenshortpl but all caps.

\glsxtrposthyphenshortpl { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 157

As \glsxtrposthyphenshort but plural.

\GLSxtrposthyphensubsequent { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 157

As \glsxtrposthyphensubsequent but all caps.

\glsxtrposthyphensubsequent { ⟨*entry-label*⟩ } { ⟨*insert*⟩ }
glossaries−extra v1.17+ — 157

Used within the post-link hook to format the insert according to the long−hyphen−postshort−hyphen style on subsequent use.

\glsxtrpostlink — §5.5.4; 255

A post-link hook that does `\glsxtrpostlink`⟨*category*⟩ if that command has been defined, where the category label is obtained from the entry that has just been referenced with a `\gls`-like or `\glstext`-like command (using `\glslabel`). Does nothing if `\glsxtrpostlink`⟨*category*⟩ isn't defined.

\glsxtrpostlinkAddDescOnFirstUse — §5.5.4; 256

May be used within a post-link hook to display the description in parentheses on first use.

\glsxtrpostlinkAddSymbolDescOnFirstUse glossaries−extra v1.31+ — §5.5.4; 256

May be used within a post-link hook to display the symbol and description in parentheses on first use.

\glsxtrpostlinkAddSymbolOnFirstUse — §5.5.4; 256

May be used within a post-link hook to display the symbol in parentheses on first use.

\glsxtrpostlink⟨*category*⟩

The post-link hook associated with the category identified by the label ⟨*category*⟩.

\glsxtrpostlinkendsentence — §5.5.4; 253

A post-link hook that's used if a full stop is discarded in order to adjust the space factor (to denote the end of a sentence). If the category post-link hook exists, and will be applied and the full stop will be restored.

\glsxtrpostlinkhook §5.5.4; 252

A post-link hook that checks if a following full stop needs to be discarded, in which case it does `\glsxtrpostlinkendsentence`, otherwise it does `\glsxtrpostlink`.

\glsxtrpostlinkSymbolDescSep glossaries−extra v1.49+ §5.5.4; 256

Used by `\glsxtrpostlinkAddSymbolDescOnFirstUse` to separate the symbol and description, if both are set.

\glsxtrpostlocalreset{⟨*entry-label*⟩} §5.10; 289

Hook performed by `\glslocalreset`. This hook is modified by `\glsenableentrycount` and `\glsenableentryunitcount`.

\glsxtrpostlocalunset{⟨*entry-label*⟩} §5.10; 289

Hook performed by `\glslocalunset`. This hook is modified by `\glsenableentrycount` and `\glsenableentryunitcount`.

\glsxtrpostlongdescription glossaries−extra v1.12+ §3.1; 33

Hook added to the end of the `description` field by the unstarred version of `\longnewglossaryentry`.

\glsxtrpostname⟨*category*⟩ glossaries−extra v1.04+ §8.6.1; 437

The post-name hook associated with the category identified by the label ⟨*category*⟩.

\glsxtrpostnamehook{⟨*entry-label*⟩} §8.6.1; 436

A hook that's performed within `\glossentryname` and `\glossentrynameother` after the entry name is displayed. This hook implements auto-indexing (see §12), then the general hook `\glsextrapostnamehook` and finally the `\glsxtrpostname`⟨*category*⟩ hook.

\GlsXtrPostNewAbbreviation §4.5.3.1; 170

A hook that's performed after the entry has been defined.

\glsxtrpostreset{⟨*entry-label*⟩} §5.10; 289

Hook performed by `\glsreset`. This hook is modified by `\glsenableentrycount` and `\glsenableentryunitcount`.

\glsxtrpostunset{⟨*entry-label*⟩} §5.10; 289

Hook performed by `\glsunset`. This hook is modified by `\glsenableentrycount` and `\glsenableentryunitcount`.

\glsxtrpostuserlongformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 150

Formats the long form in parentheses (with `\glsxtruserparen`) in styles like short−postlong−user.

\glsxtrpostusershortformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 149

Formats the short form in parentheses (with `\glsxtruserparen`) in styles like long−postshort−user.

\GlsXtrPrefixLabelFallbackLastfalse glossaries−extra−bib2gls v1.49+ §11.6.7; 613

Sets the `\ifGlsXtrPrefixLabelFallbackLast` conditional to false.

\GlsXtrPrefixLabelFallbackLasttrue glossaries−extra−bib2gls v1.49+ §11.6.7; 613

Sets the `\ifGlsXtrPrefixLabelFallbackLast` conditional to true.

\glsxtrpreglossarystyle glossaries−extra v1.49+ §8.6; 434

Hook performed by \setglossarystyle to initialise default definitions of style commands.

\glsxtrprelocation *initial:* \space glossaries−extra−stylemods v1.21+ §8.6.5; 444

Used before the location list in the predefined styles provided by glossaries−extra or modified by glossaries−extra−stylemods.

\glsxtrprenamehook{⟨*entry-label*⟩} glossaries−extra v1.54+ §8.6.1; 436

A hook that's performed within \glossentryname and \glossentrynameother before the entry name is displayed (but after the abbreviation style is set for the entry's category, if applicable). Does nothing by default.

\glsxtrprependlabelprefix{⟨*label-prefix*⟩} glossaries−extra−bib2gls v1.37+ §11.6.7; 612

Prepends ⟨*label-prefix*⟩ to the list of known labels.

\GlsXtrPreToDefaultGlsOpts{⟨*options*⟩} glossaries−extra v1.49+ §5.1.1; 191

Locally prepend ⟨*options*⟩ to the default options for the \gls-like and \glstext-like commands.

\glsxtrprovideaccsuppcmd{⟨*category*⟩}{⟨*field*⟩} glossaries−extra v1.42+
(requires accsupp)

Defines \glsxtr⟨*category*⟩⟨*field*⟩accsupp to \glsshortaccsupp, if not already defined.

\GlsXtrProvideBibTeXFields glossaries−extra−bib2gls v1.29+ §11.6.2; 583

Provides the standard BibTeX fields as glossary entry keys (using \glsaddstoragekey).

\glsxtrprovidecommand{ ⟨*cs*⟩ } [⟨*n*⟩] [⟨*default*⟩] { ⟨*definition*⟩ }
modifier: * glossaries−extra−bib2gls v1.27+

§11.6.8; 622

Just uses \providecommand within the LaTeX document but is treated as \renewcommand by bib2gls's interpreter.

\glsxtrprovidestoragekey{ ⟨*key*⟩ } { ⟨*default value*⟩ } { ⟨*no link cs*⟩ }
modifier: * glossaries−extra v1.12+

§3.2; 36

Like \glsaddstoragekey but does nothing if the key has already been defined.

\glsxtrrecentanchor glossaries−extra−bib2gls v1.49+

§11.6.6; 606

Defined by \glsxtrdisplaylocnameref to expand to the ⟨*href*⟩ argument. This corresponds to the value of \@currentHref when the record was created.

\GlsXtrRecordCount{ ⟨*entry-label*⟩ } { ⟨*counter*⟩ } glossaries−extra v1.21+

§11.5; 569

Expands to the entry's record count for the given counter (stored in the recordcount.⟨*counter*⟩ field) or to 0 if not set.

\GlsXtrRecordCounter{ ⟨*counter-name*⟩ } glossaries−extra v1.12+
(preamble only)

§8.4.3.2; 421

Activates recording for the given counter.

\glsxtrrecordtriggervalue{ ⟨*entry-label*⟩ } glossaries−extra v1.21+

§11.5; 570

Expands to the trigger value used by \glsxtrifrecordtrigger.

\GlsXtrRecordWarning{ ⟨*glossary-type*⟩ } glossaries−extra v1.31+

Incorrect use of \printglossary with non-hybrid record.

\glsxtrregularfont { ⟨*text*⟩ } glossaries−extra v1.04+ §5.5.2; 245

Used by \glsentryfmt to encapsulate regular entries. Also used by \glsxtrassignfieldfont for regular entries.

\GlsXtrResetLocalBuffer glossaries−extra v1.49+ §5.10.1; 294

If local unset for repeat entries has been enabled with \GlsXtrUnsetBufferEnableRepeatLocal, this will locally reset all entries that are in the buffer that hadn't been marked as used before the function was enabled.

\glsxtrresourcecount glossaries−extra v1.12+ §11; 542

A count register that is incremented on each use of \GlsXtrLoadResources to provide a unique basename for each resource set.

\glsxtrresourcefile [⟨*options*⟩] { ⟨*basename*⟩ } glossaries−extra v1.11+

For use with bib2gls, this both sets up the options for the resource set (which bib2gls can detect from the aux file) and inputs the file ⟨*basename*⟩**.**glstex file created by bib2gls. This command is now deprecated. Use \glsbibdata instead.

\glsxtrresourceinit glossaries−extra v1.21+ §11; 543

May be defined to temporarily change command definitions before information is written to the aux file by the protected write used by \GlsXtrLoadResources.

\GlsXtrResourceInitEscSequences glossaries−extra−bib2gls v1.51+ §11.6.2; 582

May be added to the definition of \glsxtrresourceinit to temporarily change the definitions of commands that may be used in regular expressions or within the assign-fields resource option.

\glsxtrrestoremarkhook §5.3.3; 231

Counteracts \glsxtrmarkhook.

\glsxtrrestorepostpunc glossaries−extra v1.23+ §8.6.2; 439

Used in the description to counteract the use of \glsxtrnopostpunc. Does nothing outside of the glossary.

\glsxtrrevert{⟨*text*⟩} glossaries−extra v1.49+ 177

Style-sensitive abbreviation command designed to counteract any font change applied by the style.

\glsxtrRevertMarks §5.3; 205

Restores \markright, \markboth and \@starttoc to their previous definitions.

\glsxtrRevertTocMarks glossaries−extra v1.31+ §5.3; 205

Restores \@starttoc to its previous definition.

\glsxtrsaveinsert{⟨*entry-label*⟩}{⟨*insert*⟩} glossaries−extra v1.49+ §5.5.4; 258

Implemented at the start of all the \glstext-like commands (except the inline full form commands like \glsxtrfull) to save the \glsinsert placeholder. By default, this sets \glsinsert to empty.

\glsxtrscfont{⟨*text*⟩}

Maintained for backwards-compatibility used to typeset ⟨*text*⟩ in small capitals (\textsc) for the "sc" abbreviation styles.

\glsxtrsconlydescname glossaries−extra v1.48+ 161

Expands to the name value for styles like long−only−short−sc−only−desc.

\glsxtrsconlydescsort glossaries−extra v1.48+ 162

Expands to the sort value for styles like long−only−short−sc−only−desc.

\glsxtrsconlyname glossaries−extra v1.48+ 161

Expands to the name value for styles like long−only−short−sc−only.

\glsxtrsconlyrevert{⟨*text*⟩} glossaries−extra v1.49+ 160

The definition of \glsxtrrevert used by styles like long−only−short−sc−only. Uses \glsxtrscrevert.

\glsxtrsconlysuffix *initial:* \glsxtrscsuffix
glossaries−extra v1.48+ 160

The plural suffix used by the "sc-only" abbreviation styles (such as long−only−short−sc−only).

\glsxtrscrevert{⟨*text*⟩} glossaries−extra v1.49+ 162

The definition of \glsxtrrevert used by the small caps ("sc") abbreviation styles. Uses \glstextup.

\glsxtrscsuffix 162

The plural suffix used by the small caps ("sc") abbreviation styles. This switches off the small caps font to prevent the suffix from also appearing in small caps.

\glsxtrscuserrevert{⟨*text*⟩} glossaries−extra v1.49+ 143

The definition of \glsxtrrevert used by styles like long−postshort−sc−user. Uses \glsxtrscrevert.

\glsxtrscusersuffix glossaries−extra v1.48+ 143

The plural suffix used by styles like long−postshort−sc−user.

\glsxtrseealsolabels{⟨*entry-label*⟩} glossaries−extra v1.16+ §5.9.2; 286

Expands to the value of the seealso field for the entry identified by ⟨*entry-label*⟩. If the field isn't set, this will expand to nothing. If the entry isn't defined, this will expand to \relax.

\glsxtrseeitemformat{⟨*entry-label*⟩} glossaries−extra v1.9+ §5.13; 308

Provided for use in a redefinition of \glsseeitemformat if a hierarchical name is preferred.

\glsxtrseelist{⟨*csv-list*⟩} glossaries−extra v1.16+ §5.13; 307

Fully expands ⟨*csv-list*⟩ and passes it to \glsseelist.

\glsxtrseelists{⟨*entry-label*⟩} glossaries−extra v1.49+ §5.9.2; 285

If the entry given by ⟨*entry-label*⟩ has the see, seealso or alias fields set, this will display the cross reference according to \glsxtruseseeformat (for see and alias) or \glsxtruseseealsoformat (for seealso). If any of these fields are set, the list is encapsulated with \glsxtrseelistsencap.

\glsxtrseelistsdelim glossaries−extra v1.49+ §5.9.2; 286

Used by \glsxtrseelists to as separator between sub-lists.

\glsxtrseelistsencap{⟨*content*⟩} glossaries−extra v1.49+ §5.9.2; 285

Used by \glsxtrseelists to encapsulate the lists.

\glsxtrsetactualanchor{⟨*counter*⟩} glossaries−extra−bib2gls v1.49+ §11.6.6; 606

Hook used by \glsxtrdisplaylocnameref to override the default definition of \glsxtractualanchor.

\GlsXtrSetActualChar{⟨*character*⟩} (preamble only) §12; 627

Sets the "actual character" for the auto-indexing feature.

\glsxtrsetaliasnoindex glossaries−extra v1.12+ §5.9.3; 287

Hook used to switch off indexing for aliases.

\GlsXtrSetAltModifier{⟨*token*⟩}{⟨*options*⟩} §5; 188

Sets ⟨*token*⟩ as a modifier for the `\gls`-like and `\glstext`-like commands that will automatically implement the given options.

\glsxtrsetbibglsaux{⟨*basename*⟩} glossaries−extra v1.49+ §2.4; 27
(requires `bib2gls` v3.0+)

As the `bibglsaux` option.

\glsxtrsetcategory{⟨*entry-labels*⟩}{⟨*category-label*⟩} §10; 524

Globally sets the `category` field to the fully expanded ⟨*category-label*⟩ for each entry listed in ⟨*entry-labels*⟩.

\glsxtrsetcategoryforall{⟨*glossary-labels*⟩}{⟨*category-label*⟩} §10; 524

Globally sets the `category` field to the fully expanded ⟨*category-label*⟩ for each entry belonging to the glossaries listed in ⟨*glossary-labels*⟩.

\glsxtrsetcomplexstyle{⟨*entry-label*⟩}{⟨*n*⟩} glossaries−extra v1.49+ §4.5.3.1; 172

Indicates that the entry given by ⟨*entry-label*⟩ uses a complex abbreviation style. The second argument ⟨*n*⟩ should be numeric, which indicates why it doesn't work with the variations of `\glsfirst`: 1 (all caps doesn't work), 2 (all caps and insert doesn't work), 3 (insert doesn't work).

\GlsXtrSetDefaultGlsOpts{⟨*options*⟩} §5.1.1; 191

Locally set the default options for the `\gls`-like and `\glstext`-like commands.

\GlsXtrSetDefaultNumberFormat{⟨*encap*⟩} glossaries−extra v1.19+ §5.1.1; 192

Sets the default `format` to ⟨*encap*⟩ (without the leading backslash).

\GlsXtrSetDefaultRangeFormat{⟨*encap*⟩} glossaries−extra v1.50+ §5.8; 267

Sets the default `format` to ⟨*encap*⟩ (without the leading backslash) for `\glsstartrange` and `\glsendrange`.

\GlsXtrSetEncapChar{⟨*character*⟩} (preamble only) §12; 627

Sets the "encap character" for the auto-indexing feature.

\GlsXtrSetEscChar{⟨*character*⟩} (preamble only) §12; 627

Sets the "escape character" for the auto-indexing feature.

\GlsXtrSetField{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*value*⟩} glossaries−extra v1.12+ §3.5; 40

Assigns ⟨*value*⟩ to the field identified by its internal label ⟨*field-label*⟩ for the entry identified by ⟨*entry-label*⟩. An error (or warning with `undefaction=warn`) occurs if the entry hasn't been defined.

\glsxtrsetfieldifexists{⟨*entry-label*⟩}{⟨*field-label*⟩}{⟨*code*⟩} glossaries−extra v1.12+ §3.5; 40

Used by commands like `\GlsXtrSetField` to check if the entry exists before assigning a value to the field. The ⟨*code*⟩ part is the assignment code, which is only done if the required condition is met. This can be redefined if the condition needs to be altered.

\glsxtrsetgrouptitle{⟨*group-label*⟩}{⟨*group-title*⟩} glossaries−extra v1.14+ §8.6.4; 442

Globally assigns the given title ⟨*group-title*⟩ to the group identified by ⟨*group-label*⟩.

\glsxtrsetglossarylabel{⟨*label*⟩} glossaries−extra v1.39+ §8.3; 387

Sets the label to add (using `\label{⟨label⟩}`) after the glossary section heading.

\GlsXtrSetLevelChar{⟨*character*⟩} (preamble only) §12; 627

Sets the "level character" for the auto-indexing feature.

\glsxtrsetlongfirstuse{⟨*entry-label*⟩} glossaries v1.49+ §4.3; 53

Implemented by the `\glsxtrlong` set of commands to assign `\glsxtrifwasfirstuse`.

\GlsXtrSetPlusModifier{⟨*options*⟩} glossaries−extra v1.49+ §5; 188

Overrides the options that should be implemented by the plus (+) modifier for `\gls`-like and `\glstext`-like commands.

\glsxtrsetpopts{⟨*options*⟩} glossaries−extra v1.07+ §5.4; 235

Sets the options that `\glsxtrp` (and case-change variants) pass to the relevant `\glstext`-like command.

\glsxtrsetpunctuationmarks{⟨*token list*⟩} §5.5.4; 255

Sets the punctuation list used by `\glsxtrifnextpunc`. The ⟨*token list*⟩ must be a non-delimited list of single tokens that represent each punctuation character. Note that the element of the list must be a single token, which means a single-byte character for pdfLaTeX (for example, ASCII). Multi-byte characters (UTF-8) will required a native Unicode engine (XeLaTeX or LuaLaTeX).

\GlsXtrSetRecordCountAttribute{ ⟨*category-list*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.21+

§11.5; 570

Sets the `recordcount` attribute to ⟨*value*⟩ for each of the listed categories.

\GlsXtrSetStarModifier{ ⟨*options*⟩ } glossaries−extra v1.49+

§5; 188

Overrides the options that should be implemented by the star (*) modifier for `\gls`-like and `\glstext`-like commands.

\glsxtrsetupfulldefs

§4.3; 55

Hook used by `\glsxtrfull` (and case-changing and plural variations).

\glsxtrSetWidest{ ⟨*type*⟩ } { ⟨*level*⟩ } glossaries−extra−bib2gls v1.37+

§11.6.3; 602

Written to the `glstex` by the `set-widest` option.

\glsxtrSetWidestFallback{ ⟨*type*⟩ } { ⟨*level*⟩ } glossaries−extra−bib2gls v1.37+

§11.6.3; 603

Written to the `glstex` by the `set-widest` option if `bib2gls` can't determine the widest name.

\GLSxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩

§4.3; 53

As `\glsxtrshort` but converts the link text to all caps.

\Glsxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩

§4.3; 53

As `\glsxtrshort` but converts the first character of the link text to uppercase (for the start of a sentence) using `\makefirstuc`.

\glsxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩

§4.3; 51

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the `short` value, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glslink` options.

\glsxtrshortdescname glossaries−extra v1.17+

153

Expands to the name value for styles like short−nolong−desc.

\GLSxtrshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

182

As `\glsxtrshortformat` but all caps.

\GlsxtrshortFormat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

182

As `\glsxtrshortformat` but sentence case.

\glsxtrshortFormat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

182

Encapsulates the `short` field for the given entry with ⟨*fmt-cs*⟩. The ⟨*insert*⟩ argument is the insertion material supplied in the final optional argument of the `\gls`-like or `\glstext`-like commands. The `\ifglsxtrinsertinside`, inner formatting, and accessibility settings are supported.

\GLSxtrshortformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+

183

As `\glsxtrshortformatgrp` but all caps.

\Glsxtrshortformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 183

As \glsxtrshortformatgrp but sentence case.

\glsxtrshortformatgrp{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*fmt-cs*⟩}
glossaries−extra v1.49+ 183

As \glsxtrshortformat but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\glsxtrshorthyphen{⟨*short*⟩}{⟨*entry-label*⟩}{⟨*insert*⟩} glossaries−extra v1.17+ 158

Formats the short form according to the short−hyphen−postlong−hyphen style.

\GLSxtrshorthyphenlong{⟨*entry-label*⟩}{⟨*short*⟩}{⟨*long*⟩}{⟨*insert*⟩}
glossaries−extra v1.49+ 158

As \glsxtrshorthyphenlong but ⟨*insert*⟩ is converted to all caps.

\glsxtrshorthyphenlong{⟨*entry-label*⟩}{⟨*short*⟩}{⟨*long*⟩}{⟨*insert*⟩}
glossaries−extra v1.17+ 158

Formats the short and long form according to the short−hyphen−long−hyphen style.

\glsxtrshorthyphenlongsort glossaries−extra v1.49+ 154

Expands to the sort value for the short−hyphen−long−hyphen styles.

\glsxtrshortlongdescname glossaries−extra v1.17+ 141

Expands to the name value for short−long−desc styles.

\glsxtrshortlongdescsort glossaries−extra v1.17+ 141

Expands to the sort value for short−long−desc styles.

\GLSxtrshortlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 185

As \glsxtrshortlongformat but all caps.

\Glsxtrshortlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 185

As \glsxtrshortlongformat but sentence case.

\glsxtrshortlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 184

Formats the short form with \glsxtrshortformat and the long form in parentheses with \glsxtrlongformat.

\glsxtrshortlongname glossaries−extra v1.25+ 141

Expands to the name value for short−long styles.

\GLSxtrshortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 185

As \glsxtrshortlongplformat but all caps.

\Glsxtrshortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 185

As \glsxtrshortlongplformat but sentence case.

\glsxtrshortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 185

As \glsxtrshortlongformat but for the plurals.

\glsxtrshortlonguserdescname glossaries−extra v1.25+ 145

Expands to the value for the name key for styles like short−long−user−desc.

\glsxtrshortnolongname glossaries−extra v1.25+ 152

Expands to the name value for short−nolong styles.

\GLSxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 53

As \glsxtrshort but converts the link text to all caps.

\Glsxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 53

As \glsxtrshortpl but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glsxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ §4.3; 53

References the entry identified by ⟨*entry-label*⟩. The text produced is obtained from the short-plural value, formatted according to the abbreviation style associated with the entry's category. The ⟨*insert*⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\GLSxtrshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 182

As \glsxtrshortplformat but all caps.

\Glsxtrshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 182

As \glsxtrshortplformat but sentence case.

\glsxtrshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 182

As \glsxtrshortformat but for the shortplural field.

\GLSxtrshortplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 183

As \glsxtrshortplformatgrp but all caps.

\Glsxtrshortplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 183

As \glsxtrshortplformatgrp but sentence case.

\glsxtrshortplformatgrp { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*fmt-cs*⟩ }
glossaries−extra v1.49+ 183

As \glsxtrshortplformat but adds grouping around ⟨*insert*⟩ (with the inner formatting inside the group).

\glsxtrshowtargetinner { ⟨*target-name*⟩ } glossaries−extra v1.48+ §2.5; 31

Used in inner mode for debugging, this defaults to \glsshowtargetinner but is changed by the showtargets options.

\glsxtrshowtargetouter { ⟨*target-name*⟩ } glossaries−extra v1.48+ §2.5; 30

Used in outer mode for debugging, this defaults to \glsshowtargetouter but is changed by the showtargets options.

\glsxtrshowtargetsymbolleft glossaries−extra v1.48+ §2.5; 31

The left marker debugging symbol (▹).

\glsxtrshowtargetsymbolright glossaries−extra v1.48+ §2.5; 31

The right marker debugging symbol (◃).

\glsxtrsmfont{⟨*text*⟩}

Maintained for backwards-compatibility used to typeset ⟨*text*⟩ in a smaller font (`\textsmaller`) for the "sm" abbreviation styles.

\glsxtrsmrevert{⟨*text*⟩} glossaries−extra v1.49+ 163

The definition of `\glsxtrrevert` used by the smaller ("sm") abbreviation styles. Uses `\textlarger`.

\glsxtrsmsuffix *initial:* `\glsxtrabbrvpluralsuffix` 163

The plural suffix used by the smaller ("sm") abbreviation styles (such as short−sm−long).

\glsxtrspacerules glossaries−extra−bib2gls v1.27+ 590

Expands to space character sort rules.

\GlsXtrStandaloneEntryHeadName{⟨*entry-label*⟩} glossaries−extra v1.49+ §8.5; 432

Used by `\glsxtrglossentry` for the header and toc.

\GlsXtrStandaloneEntryHeadNameFirstUc{⟨*entry-label*⟩}
glossaries−extra v1.54+ §8.5; 433

Used by `\Glsxtrglossentry` for the header and toc.

\GlsXtrStandaloneEntryHeadOther{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.49+ §8.5; 433

Used by \glsxtrglossentryother for the header and toc.

\GlsXtrStandaloneEntryHeadOtherFirstUc{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.54+ §8.5; 434

Used by \Glsxtrglossentryother for the header and toc.

\GlsXtrStandaloneEntryName{⟨*entry-label*⟩} glossaries−extra v1.37+ §8.5; 430

Used by \glsxtrglossentry to display the standalone entry name and create the associated hypertarget, if supported.

\GlsXtrStandaloneEntryNameFirstUc{⟨*entry-label*⟩}
glossaries−extra v1.54+ §8.5; 430

Used by \Glsxtrglossentry to display the standalone entry name and create the associated hypertarget, if supported.

\GlsXtrStandaloneEntryOther{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.37+ §8.5; 431

As \GlsXtrStandaloneEntryName but where the text is obtained from the given field instead of name.

\GlsXtrStandaloneEntryOtherFirstUc{⟨*entry-label*⟩}{⟨*field-label*⟩}
glossaries−extra v1.54+ §8.5; 431

As \Glsxtrglossentryother to produce the sentence case field text.

\GlsXtrStandaloneEntryPdfName{⟨*entry-label*⟩} glossaries−extra v1.49+ §8.5; 432

Used by \glsxtrglossentry for the PDF bookmark.

\GlsXtrStandaloneEntryPdfNameFirstUc{ ⟨*entry-label*⟩ }
glossaries−extra v1.54+ §8.5; 433

Used by \Glsxtrglossentry for the PDF bookmark.

\GlsXtrStandaloneEntryPdfOther{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.49+ §8.5; 433

Used by \glsxtrglossentryother for the PDF bookmark.

\GlsXtrStandaloneEntryPdfOtherFirstUc{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
glossaries−extra v1.54+ §8.5; 433

Used by \Glsxtrglossentryother for the PDF bookmark.

\GlsXtrStandaloneGlossaryType glossaries−extra v1.21+ §8.5; 429

Expands to the glossary type for standalone entries.

\GlsXtrStandaloneSubEntryItem{ ⟨*entry-label*⟩ } glossaries−extra v1.21+ §8.5; 429

Used to display standalone entries that have the parent field set.

\glsxtrstarflywarn

Issues a warning with \GlossariesExtraWarning indicating that the experimental starred version of \GlsXtrEnableOnTheFly has been used.

\GlsXtrStartUnsetBuffering *modifier:* * glossaries−extra v1.30+ §5.10.1; 293

Enables unset buffering. The starred version doesn't check for duplicates.

\GlsXtrStopUnsetBuffering *modifier:* * glossaries−extra v1.30+ §5.10.1; 293

Stops buffering. The starred version performs a global unset.

\glsxtrSubScriptDigitrules glossaries−extra−bib2gls v1.27+ 595

Expands to the 0–9 subscript digit character sort rules.

\GLSxtrsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 179

Used by `\glsxtrgenabbrvfmt` to display the all caps subsequent singular form (defined by the abbreviation style).

\Glsxtrsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 178

Used by `\glsxtrgenabbrvfmt` to display the sentence case subsequent singular form (defined by the abbreviation style).

\glsxtrsubsequentfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 178

Used by `\glsxtrgenabbrvfmt` to display the subsequent singular form (defined by the abbreviation style).

\GLSxtrsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.49+ 179

Used by `\glsxtrgenabbrvfmt` to display the all caps subsequent plural form (defined by the abbreviation style).

\Glsxtrsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 178

Used by `\glsxtrgenabbrvfmt` to display the sentence case subsequent plural form (defined by the abbreviation style).

\glsxtrsubsequentplfmt { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.17+ 178

Used by `\glsxtrgenabbrvfmt` to display the subsequent plural form (defined by the abbreviation style).

\glsxtrSuperScriptDigitrules glossaries−extra−bib2gls v1.27+ 595

Expands to the 0–9 superscript digit character sort rules.

\glsxtrsupphypernumber{⟨*location*⟩} glossaries−extra v1.14+ §5.1.2; 199

Used to hyperlink to a location in an external document if the `externallocation` attribute has been set. This will define `\glsxtrsupplocationurl` to the location provided by the attribute or to empty if the attribute isn't set.

\glsxtrsupplocationurl glossaries−extra v1.14+

Defined by `\glsxtrsupphypernumber` to the external location or empty if not provided.

\glsxtrtagfont{⟨*text*⟩} §4.4; 59

Used by the tagging command defined with `\GlsXtrEnableInitialTagging`.

\glsxtrtaggedlist{⟨*singular tag*⟩}{⟨*plural tag*⟩}{⟨*label prefix*⟩}{⟨*csv-list*⟩} glossaries−extra v1.49+ §5.13; 308

Similar to `\glsseelist`, this will start the list with ⟨*singular tag*⟩ if the list only contains one element and ⟨*plural tag*⟩ if the list contains more than one element. Each element is prefixed with ⟨*label prefix*⟩. The tag is separated from the start of the list with `\glsxtrtaggedlistsep`. The actual list separators as as for `\glsseelist`. The ⟨*csv-list*⟩ is expanded before being iterated over. Does nothing if ⟨*csv-list*⟩ is empty.

\glsxtrtaggedlistsep *initial:* `\space` glossaries−extra v1.49+ §5.13; 308

Separator used by `\glsxtrtaggedlist` between the tag and the list.

\glsxtrtarget{⟨*entry-label*⟩}{⟨*text*⟩} glossaries−extra v1.51+ §5.6; 262

Like `\glstarget` but only creates the target if the field given by `\glsxtrtarget` field

hasn't been set (if hyperlinks are supported). If that field hasn't been set, the target is created and the field is set to the target name. Otherwise, `\glsxtrtargetdup` will be used.

\glsxtrtargetdup{⟨*entry-label*⟩}{⟨*text*⟩} glossaries−extra v1.54+ §5.6; 262

Behaviour of `\glsxtrtarget` if the target has already be set.

\glsxtrtargetfield *initial:* `target` glossaries−extra v1.51+ §5.6; 262

Expands to the field label used by `\glsxtrtarget`.

\GlsXtrTheLinkCounter{⟨*entry-label*⟩} glossaries−extra v1.26+ §6.2; 328

Expands to the value of the link counter associated with the given entry or 0 if it hasn't been defined.

\glsxtrtitlednamereflink{⟨*format*⟩}{⟨*location*⟩}{⟨*title*⟩}{⟨*file*⟩}
glossaries−extra−bib2gls v1.49+ §11.6.6; 608

Used by `\glsxtrdisplaylocnameref` to display locations that have a title and are not associated with the page counter and don't have an associated `\glsxtr`⟨*counter*⟩`locfmt` command. The anchor is obtained from `\glsxtrrecentanchor`.

\GLSxtrtitlefirst{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 229

Used to display the all caps entry's `first` field in the section title and table of contents.

\Glsxtrtitlefirst{⟨*entry-label*⟩} §5.3.3; 229

Used to display the sentence case entry's `first` field in the section title and table of contents.

\glsxtrtitlefirst{⟨*entry-label*⟩} §5.3.3; 229

Used to display the entry's `first` field in the section title and table of contents.

\GLSxtrtitlefirstplural{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 230

Used to display the all caps entry's `firstplural` field in the section title and table of contents.

\Glsxtrtitlefirstplural{⟨*entry-label*⟩} §5.3.3; 229

Used to display the sentence case entry's `firstplural` field in the section title and table of contents.

\glsxtrtitlefirstplural{⟨*entry-label*⟩} §5.3.3; 229

Used to display the entry's `firstplural` field in the section title and table of contents.

\GLSxtrtitlefull{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 226

Used to display the entry's all caps full form in the section title and table of contents.

\Glsxtrtitlefull{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

Used to display the entry's sentence case full form in the section title and table of contents.

\glsxtrtitlefull{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

Used to display the entry's full form in the section title and table of contents.

\GLSxtrtitlefullpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 226

Used to display the entry's all caps full plural form in the section title and table of contents.

\Glsxtrtitlefullpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 226

Used to display the entry's sentence case full plural form in the section title and table of contents.

\glsxtrtitlefullpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 226

Used to display the entry's full plural form in the section title and table of contents.

\GLSxtrtitlelong{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 224

The normal behaviour of `\GLSfmtlong`.

\Glsxtrtitlelong{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 224

The normal behaviour of `\Glsfmtlong`.

\glsxtrtitlelong{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 224

The normal behaviour of `\glsfmtlong`.

\GLSxtrtitlelongpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 225

The normal behaviour of `\GLSfmtlongpl`.

\Glsxtrtitlelongpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

The normal behaviour of `\Glsfmtlongpl`.

\glsxtrtitlelongpl{⟨*entry-label*⟩} glossaries−extra v1.02+ §5.3.3; 225

The normal behaviour of `\glsfmtlongpl`.

\GLSxtrtitlename{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 227

Used to display the all caps entry's name in the section title and table of contents.

\Glsxtrtitlename{ ⟨*entry-label*⟩ } glossaries−extra v1.21+ §5.3.3; 227

Used to display the sentence case entry's name in the section title and table of contents.

\glsxtrtitlename{ ⟨*entry-label*⟩ } glossaries−extra v1.21+ §5.3.3; 227

Used to display the entry's name in the section title and table of contents.

\glsxtrtitleopts glossaries−extra v1.49+ §5.3.2; 210

Expands to the options that commands like `\glsfmtshort` should use in the title or caption within the document text.

\glsxtrtitleorpdforheading{ ⟨*title*⟩ } { ⟨*PDF bookmarks*⟩ } { ⟨*heading*⟩ }
glossaries−extra v1.21+ §5.3.3; 218

Does the applicable argument depending on whether the command occurs within a title/caption or PDF bookmark or heading.

\GLSxtrtitleplural{ ⟨*entry-label*⟩ } glossaries−extra v1.42+ §5.3.3; 228

Used to display the all caps entry's `plural` field in the section title and table of contents.

\Glsxtrtitleplural{ ⟨*entry-label*⟩ } §5.3.3; 228

Used to display the sentence case entry's `plural` field in the section title and table of contents.

\glsxtrtitleplural{ ⟨*entry-label*⟩ } §5.3.3; 228

Used to display the entry's `plural` field in the section title and table of contents.

\GLSxtrtitleshort{ ⟨*entry-label*⟩ } glossaries−extra v1.42+ §5.3.3; 222

The normal behaviour of `\GLSfmtshort`.

\Glsxtrtitleshort{⟨*entry-label*⟩} §5.3.3; 222

The normal behaviour of `\Glsfmtshort`.

\glsxtrtitleshort{⟨*entry-label*⟩} §5.3.3; 221

The normal behaviour of `\glsfmtshort`.

\GLSxtrtitleshortpl{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 224

The normal behaviour of `\GLSfmtshortpl`.

\Glsxtrtitleshortpl{⟨*entry-label*⟩} §5.3.3; 223

The normal behaviour of `\Glsfmtshortpl`.

\glsxtrtitleshortpl{⟨*entry-label*⟩} glossaries−extra v1.03+ §5.3.3; 223

The normal behaviour of `\glsfmtshortpl`.

\GLSxtrtitletext{⟨*entry-label*⟩} glossaries−extra v1.42+ §5.3.3; 228

Used to display the all caps entry's `text` field in the section title and table of contents.

\Glsxtrtitletext{⟨*entry-label*⟩} §5.3.3; 227

Used to display the sentence case entry's `text` field in the section title and table of contents.

\glsxtrtitletext{⟨*entry-label*⟩} §5.3.3; 227

Used to display the entry's `text` field in the section title and table of contents.

\GlsXtrTotalRecordCount{ ⟨*entry-label*⟩ } glossaries−extra v1.21+ §11.5; 569

Expands to the entry's total record count (stored in the `recordcount` field) or to 0 if not set.

\glsxtrtreechildpredesc *initial:* `\glstreechildpredesc`
glossaries−extra−stylemods v1.46+

Inserted before the child descriptions for the tree styles.

\glsxtrtreepredesc *initial:* `\glstreepredesc`
glossaries−extra−stylemods v1.46+

Inserted before the top-level descriptions for the tree styles.

\glsxtrundefaction{ ⟨*message*⟩ } { ⟨*additional help*⟩ } glossaries−extra v1.08+ §2.4; 17

Will either produce an error or a warning, depending on the `undefaction` setting. In the document environment this will also generate the unknown marker (`??`).

\glsxtrundeftag *initial:* `??` glossaries−extra v1.08+ §2.4; 16

Expands to the unknown marker (`??`).

\GlsXtrUnknownDialectWarning{ ⟨*locale*⟩ } { ⟨*root language*⟩ }
glossaries−extra v1.32+ §5.12.1; 302

Issues a warning with `\GlossariesExtraWarning` indicating that a valid dialect label can't be determined for the given locale and root language.

\GlsXtrUnsetBufferDisableRepeatLocal glossaries−extra v1.49+ §5.10.1; 294

Disables GlsXtrUnsetBufferEnableRepeatLocal.

\GlsXtrUnsetBufferEnableRepeatLocal glossaries−extra v1.49+ §5.10.1; 294

Allows repeat entries within the buffering code to be locally unset before the link text.

\glsxtrunsrtdo{⟨*entry-label*⟩} glossaries−extra v1.12+ §8.4.3; 416

Used by the "unsrt" family of commands, this displays the glossary entry according to the current glossary style (taking the hierarchical level into account, which may have been adjusted by `leveloffset` or `flatten`).

\glsxtrunusedformat{⟨*location*⟩} §5.9.3; 288

The format used by `\glsxtraddallcrossrefs`.

\glsxtrUpAlpha glossaries−extra−bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright alpha.

\glsxtrUpBeta glossaries−extra−bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright beta.

\glsxtrUpChi glossaries−extra−bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright chi.

\glsxtrUpDelta glossaries−extra−bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright delta.

\glsxtrUpDigamma glossaries−extra−bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright digamma.

\glsxtrUpEpsilon glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright epsilon.

\glsxtrUpEta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright eta.

\glsxtrUpGamma glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright gamma.

\glsxtrUpIota glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright iota.

\glsxtrUpKappa glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright kappa.

\glsxtrUpLambda glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright lambda.

\glsxtrUpMu glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright mu.

\glsxtrUpNu glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright nu.

\glsxtrUpOmega glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright omega.

\glsxtrUpOmicron glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright omicron.

\glsxtrUpPhi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright phi.

\glsxtrUpPi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright pi.

\glsxtrUpPsi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright psi.

\glsxtrUpRho glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright rho.

\glsxtrUpSigma glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright sigma.

\glsxtrUpTau glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright tau.

\glsxtrUpTheta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright theta.

\glsxtrUpUpsilon glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright upsilon.

\glsxtrUpXi glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright xi.

\glsxtrUpZeta glossaries-extra-bib2gls v1.27+ 602

(Sort rule) expands to the variations of math Greek upright zeta.

\GlsXtrUseAbbrStyleFmts{ ⟨*style-name*⟩ } §4.5.2; 165

Implements the ⟨*display definitions*⟩ code for the given abbreviation style.

\GlsXtrUseAbbrStyleSetup{ ⟨*style-name*⟩ } §4.5.2; 165

Implements the ⟨*setup*⟩ code for the given abbreviation style.

\glsxtrusealias{ ⟨*entry-label*⟩ } glossaries-extra v1.42+ §5.9.2; 286

If the entry given by ⟨*entry-label*⟩ has the `alias` field set, this will display the cross reference according to `\glsxtruseseeformat`.

\GLSxtrusefield{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } glossaries-extra v1.37+ §5.11; 300

As `\glsxtrusefield` but converts the field value to all caps.

\Glsxtrusefield{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.12+ §5.11; 299

As \glsxtrusefield but uses sentence case.

\glsxtrusefield{⟨*entry-label*⟩}{⟨*field-label*⟩} glossaries−extra v1.12+ §5.11; 299

Expands to the value of the given field (identified by its internal label ⟨*field-label*⟩) for the entry given by ⟨*entry-label*⟩. Expands to \relax if the entry or field are undefined.

\glsxtruserfield glossaries−extra v1.04+ 142

Expands to the internal label of the field used to store additional information for the "user" abbreviation styles, such as long−short−user.

\glsxtruserfieldfmt{⟨*text*⟩} glossaries−extra v1.49+ 142

Used to format the value of the field given by \glsxtruserfield within \glsxtruserparen and \GLSxtruserparen.

\GLSxtruserlongformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩} glossaries−extra v1.49+ 150

As \glsxtruserlongformat but all caps.

\glsxtruserlongformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩} glossaries−extra v1.49+ 149

Formats the singular long form in parentheses (with \glsxtruserparen) in styles like short−long−user.

\GLSxtruserlongplformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩} glossaries−extra v1.49+ 150

As \glsxtruserlongplformat but all caps.

\glsxtruserlongplformat { ⟨*entry-label*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 150

Formats the plural long form in parentheses (with `\glsxtruserparen`) in styles like short−long−user.

\GLSxtruserlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 146

As `\glsxtruserlongshortformat` but all caps.

\Glsxtruserlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 146

As `\glsxtruserlongshortformat` but sentence case.

\glsxtruserlongshortformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 145

Used by styles like long−short−user to format the long and short form.

\GLSxtruserlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 146

As `\glsxtruserlongshortplformat` but all caps.

\Glsxtruserlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 146

As `\glsxtruserlongshortplformat` but sentence case.

\glsxtruserlongshortplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 146

Used by styles like long−short−user to format the plural long and plural short form.

\GLSxtruserparen{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−extra v1.49+ 144

As \glsxtruserparen but the value of the field given by \glsxtruserfield is converted to all caps. The ⟨*text*⟩ argument should already be in all caps.

\glsxtruserparen{⟨*text*⟩}{⟨*entry-label*⟩} glossaries−extra v1.04+ 144

Used by the "user" abbreviation styles, such as long−short−user, to insert the space separator (\glsxtrfullsep) followed by the parenthetical material (\glsxtrparen) consisting of ⟨*text*⟩ and, if set, the value of the field given by \glsxtruserfield, separated by \glsxtruserparensep.

\glsxtruserparensep glossaries−extra v1.49+ 142

The separator used in the parenthetical content of \glsxtruserparen and \GLSxtruserparen.

\GLSxtrusershortformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩} glossaries−extra v1.49+ 148

As \glsxtrusershortformat but all caps.

\glsxtrusershortformat{⟨*entry-label*⟩}{⟨*fmt-cs*⟩} glossaries−extra v1.49+ 148

Formats the singular short form in parentheses (with \glsxtruserparen) in styles like long−short−user.

\GLSxtrusershortlongformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩} glossaries−extra v1.49+ 147

As \glsxtrusershortlongformat but all caps.

\GlsxtrusershortlongformatGlsxtrusershortlongformat{⟨*entry-label*⟩}{⟨*insert*⟩}{⟨*long-fmt-cs*⟩}{⟨*short-fmt-cs*⟩} glossaries−extra v1.49+ 147

As \glsxtrusershortlongformat but sentence case.

\glsxtrusershortlongformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 147

Used by styles like short−long−user to format the short and long form.

\GLSxtrusershortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 148

As \glsxtrusershortlongplformat but all caps.

\Glsxtrusershortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 148

As \glsxtrusershortlongplformat but sentence case.

\glsxtrusershortlongplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } { ⟨*long-fmt-cs*⟩ } { ⟨*short-fmt-cs*⟩ } glossaries−extra v1.49+ 147

Used by styles like short−long−user to format the plural short and plural long form.

\GLSxtrusershortplformat { ⟨*entry-label*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 148

As \glsxtrusershortplformat but all caps.

\glsxtrusershortplformat { ⟨*entry-label*⟩ } { ⟨*fmt-cs*⟩ } glossaries−extra v1.49+ 148

Formats the plural short form in parentheses (with \glsxtruserparen) in styles like long−short−user.

\glsxtrusersuffix *initial:* \glsxtrabbrvpluralsuffix glossaries−extra v1.04+ 142

The plural suffix used by styles like short−long−user.

\glsxtrusesee{⟨*entry-label*⟩} glossaries−extra v1.06+ §5.9.2; 286

If the entry given by ⟨*entry-label*⟩ has the `see` field set, this will display the cross reference according to `\glsxtruseseeformat`.

\glsxtruseseealso{⟨*entry-label*⟩} glossaries−extra v1.16+ §5.9.2; 286

If the entry given by ⟨*entry-label*⟩ has the `seealso` field set, this will display the cross reference according to `\glsxtruseseealsoformat`.

\glsxtruseseealsoformat{⟨*csv-list*⟩} glossaries−extra v1.16+ §5.9.2; 286

Formats the comma-separated list of entry labels as a “see also” cross-reference.

\glsxtruseseeformat{⟨*tag*⟩}{⟨*xr-list*⟩} glossaries−extra v1.06+

Format used by `\glsxtrusesee`. This internally uses `\glsseeformat`.

\GlsXtrWarnDeprecatedAbbrStyle{⟨*old-name*⟩}{⟨*new-name*⟩}
glossaries−extra v1.04+ §4.5.2; 166

Issues a warning with `\GlossariesExtraWarning` indicating that a deprecated abbreviation style has been used.

\GlsXtrWarning{⟨*options*⟩}{⟨*entry*⟩} §13; 629

Issues a warning with `\GlossariesExtraWarning` indicating that the given options list has been ignored by the given entry because it has already been defined.

\glsxtrword{⟨*word*⟩} glossaries−extra v1.17+ §10.2.1.1; 528

Used to mark each word by the `markwords` and `markshortwords` attributes.

\glsxtrwordsep glossaries−extra v1.17+ §10.2.1.1; 528

Used to mark word separator space by the markwords and markshortwords attributes.

\glsxtrwordsephyphen glossaries−extra v1.49+

Used to mark compound word separator hyphen by the markwords and markshortwords attributes.

\glsxtrwrglossaryhook{⟨*entry-label*⟩}

Hook implemented everytime an entry is indexed.

\glsxtrwrglossarylocfmt{⟨*location*⟩}{⟨*title*⟩} glossaries−extra−bib2gls v1.49+ §11.6.6; 607

Used by \glsxtrdisplaylocnameref to format a location where the counter is wrglossary.

\glsxtrwrglosscountermark{⟨*number*⟩} glossaries−extra v1.49+ §2.5; 30

Used to mark where the wrglossary counter is incrememented with debug=showwrgloss.

\glsxtrwrglossmark glossaries−extra v1.21+ §2.5; 30

Marker (·) used to mark write operations with debug=showwrgloss.

H

\hyperbf{⟨*location(s)*⟩} glossaries

If hyperlinks are supported this does \textbf{\glshypernumber{⟨*location(s)*⟩}} otherwise it just does \textbf{⟨*location(s)*⟩}.

I

\ifglossaryexists{ ⟨*glossary-type*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } *modifier:* *
glossaries

If the glossary given by ⟨*glossary-type*⟩ exists, this does ⟨*true*⟩, otherwise it does ⟨*false*⟩. The unstarred form treats ignored glossaries as non-existent. The starred form (v4.46+) will do ⟨*true*⟩ if ⟨*glossary-type*⟩ matches an ignored glossary.

\ifglsentryexists{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries

Does ⟨*true*⟩ if the entry given by ⟨*entry-label*⟩ exists, otherwise does ⟨*false*⟩.

\ifglsfieldcseq{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*cs-name*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v4.16+

Tests if the value of the given field is equal to the replacement text of the command given by the control sequence name ⟨*cs-name*⟩ using etoolbox's `\ifcsstrequal`. Triggers an error if the given field (identified by its internal field label) hasn't been defined. Uses `\glsdoifexists`.

\ifglsfielddefeq{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*cs*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries v4.16+

Tests if the value of the given field is equal to the replacement text of the given command ⟨*cs*⟩ using etoolbox's `\ifdefstrequal`. Triggers an error if the given field (identified by its internal field label) hasn't been defined. Uses `\glsdoifexists`.

\ifglsfieldeq{ ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*string*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries v4.16+

Tests if the value of the given field is equal to the given string using etoolbox's `\ifcsstring`. Triggers an error if the given field (identified by its internal field label) hasn't been defined. Uses `\glsdoifexists`.

\ifglsfieldvoid{ ⟨*field-label*⟩ } { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries v4.50+

An expandable test to determine if the entry is undefined or the field is undefined or empty. The ⟨*field-label*⟩ must be the field's internal label.

\ifglshaschildren{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.02+

Does ⟨*true*⟩ if the given entry has child entries otherwise does ⟨*false*⟩. Note that this has to iterate over the set of defined entries for the entry's glossary to find one that has the entry identified in its `parent` field. A more efficient approach can be achieved with `bib2gls` and the `save-child-count` resource option.

\ifglshasdesc{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.08a+

Does ⟨*true*⟩ if the entry's `description` field is set otherwise does ⟨*false*⟩.

\ifglshasdescsuppressed{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries v3.08a+

Does ⟨*true*⟩ if the entry's `description` field is just `\nopostdesc` otherwise does ⟨*false*⟩.

\ifglshasfield{ ⟨*field*⟩ } { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v4.03+

If the field identified by either its key or its internal field label ⟨*field*⟩ for the entry identified by ⟨*entry-label*⟩ is set and non-empty, this sets `\glscurrentfieldvalue` to the field value and does ⟨*true*⟩ otherwise it does ⟨*false*⟩.

\ifglshaslong{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.11a+

Does ⟨*true*⟩ if the entry's `long` field is set otherwise does ⟨*false*⟩.

\ifglshasparent{ ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.02+

Does ⟨*true*⟩ if the entry's `parent` field is set otherwise does ⟨*false*⟩.

\ifglshasshort { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.11a+

Does ⟨*true*⟩ if the entry's `short` field is set otherwise does ⟨*false*⟩.

\ifglshassymbol { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries v3.08a+

Does ⟨*true*⟩ if the entry's `symbol` field is set otherwise does ⟨*false*⟩.

\ifglsindexonlyfirst ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
glossaries v3.02+

§5.8; 273

A conditional that corresponds to the `indexonlyfirst` option.

\ifGlsLongExtraUseTabular ⟨*true*⟩`\else` ⟨*false*⟩`\fi`
initial: `\iffalse` glossary−longextra v1.37+

§8.7.2; 461

Conditional that determines whether or not to use tabular instead of longtable. If this conditional is changed, it must be changed before the style is set.

\ifglsnogroupskip ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
glossaries v3.03+

Conditional set by the `nogroupskip` option.

\ifglsresetcurrcount ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
glossaries−extra v1.49+

§6.1; 318

Conditional that determines whether or not to reset the entry counter to 0 when the first use flag is reset.

\ifglsused { ⟨*entry-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries

Does ⟨*true*⟩ if the entry has been marked as used, does ⟨*false*⟩ if the entry is marked as unused, and does neither if the entry hasn't been defined (but will generate an error or warning according to `undefaction`).

\ifglsxtrinitwrglossbefore ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iftrue glossaries−extra v1.14+

§5.1.2; 198

A conditional that indicates whether or not wrgloss=before is set.

\ifglsxtrinsertinside ⟨*true*⟩\else ⟨*false*⟩\fi *initial:* \iffalse
glossaries−extra v1.02

138

A conditional used by the predefined abbreviation styles to determine whether the ⟨*insert*⟩ part should go inside or outside of the style's font formatting commands.

\ifGlsXtrPrefixLabelFallbackLast ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iftrue glossaries−extra−bib2gls v1.49+

§11.6.7; 613

Conditional that determines whether or not to use the last label prefix as the default.

\ifglsxtrprintglossflatten ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iffalse glossaries−extra v1.49+

§8.4.3; 416

Conditional set by the flatten option.

\ifmglsused{⟨*multi-label*⟩}{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.48+

§7.7; 362

Does ⟨*true*⟩ if the given multi-entry has been marked as used, otherwise does ⟨*false*⟩.

\ifmultiglossaryentryglobal ⟨*true*⟩\else ⟨*false*⟩\fi
initial: \iffalse glossaries−extra v1.48+

§7; 335

If true, subsequent multi-entry definitions will be global.

\IfNotBibGls{⟨*LaTeX code*⟩}{⟨*bib2gls code*⟩} glossaries−extra−bib2gls v1.54+

§11.6.8; 621

Defined by glossaries−extra−bib2gls to ⟨*LaTeX code*⟩ but defined by bib2gls's interpreter to expand to ⟨*bib2gls code*⟩.

\IfTeXParserLib{ ⟨*TEX parser lib code*⟩ } { ⟨*LATEX code*⟩ }
glossaries−extra−bib2gls v1.49+

§11.6.8; 621

Defined by glossaries−extra−bib2gls to ⟨*LATEX code*⟩ but defined by the TEX parser library to expand to ⟨*TEX parser lib code*⟩.

\IN

§11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\IN`.

\indexname *initial:* `Index` (language-sensitive)

Expands to the index title.

\INTERPRET

§11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\INTERPRET`.

\Iota glossaries−extra−bib2gls v1.27+

§11.6.8; 622

Defined with `\providecommand`, this just does `\mathrm{I}`.

K

\Kappa glossaries−extra−bib2gls v1.27+

§11.6.8; 622

Defined with `\providecommand`, this just does `\mathrm{K}`.

L

\LABELIFY

§11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\LABELIFY`.

\LABELIFYLIST §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \LABELIFYLIST.

\LC §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \LC.

\LEN §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \LEN.

\letabbreviationstyle { ⟨*new style*⟩ } { ⟨*existing style*⟩ } glossaries−extra v1.04+ §4.5.3; 166

Defines a synonym for an existing abbreviation style.

\loadglsentries [⟨*type*⟩] { ⟨*filename*⟩ } glossaries

Locally assigns \glsdefaulttype to ⟨*type*⟩ and inputs ⟨*filename*⟩. If the optional argument is omitted, the default glossary is assumed. Note that if any entries with ⟨*filename*⟩ have the type key set (including implicitly in commands like \newabbreviation), then this will override the type given in the optional argument.

\longnewglossaryentry { ⟨*entry-label*⟩ } { ⟨*key=value list*⟩ } { ⟨*description*⟩ } glossaries v3.11a §3.1; 33

Defines a new glossary entry with the given label. The second argument is a comma-separated list of glossary entry keys. The third argument is the description, which may include paragraph breaks.

\longnewglossaryentry* { ⟨*entry-label*⟩ } { ⟨*key=value list*⟩ } { ⟨*description*⟩ } glossaries−extra v1.12+ §3.1; 33

Like the unstarred \longnewglossaryentry but doesn't add the \glsxtrpostlongdescription hook.

M

\MakeAcronymsAbbreviations §4.6; 186

Counteracts \RestoreAcronyms. Not recommended.

\makefirstuc{⟨*text*⟩} mfirstuc

Robust command that converts the first character of ⟨*text*⟩ to uppercase unless ⟨*text*⟩ starts with a command, in which case it will attempt to apply the case change to the first character of the first argument following the command, if the command is followed by a group. As from mfirstuc v2.08, this command internally uses \MFUsentencecase to perform the actual case-change. See the mfirstuc documentation for further details, either:

```
texdoc mfirstuc
```

or visit ctan.org/pkg/mfirstuc.

\makeglossaries[⟨*types*⟩] glossaries §8; 383

Opens the associated glossary files that need to be processed by makeindex or xindy. The optional argument is only available with glossaries-extra and is used for a hybrid approach. All glossaries (or each glossary identified in ⟨*types*⟩) should be displayed with \printglossary. If the optional argument is present, any glossaries not identified in ⟨*types*⟩ should be displayed with \printnoidxglossary.

\makenoidxglossaries glossaries v4.04+

Sets up all non-ignored glossaries so that they can be displayed with \printnoidxglossary.

\mfirstucMakeUppercase{⟨*text*⟩} mfirstuc

This command was used by \makefirstuc to convert its argument to all caps and was redefined by glossaries to use \MakeTextUppercase, but with mfirstuc v2.08+ and glossaries v4.50+ this command is instead defined to use the LaTeX3 all caps command, which is

expandable. This command is no longer used by `\makefirstuc` (which instead uses `\MFUsentencecase`) or by glossaries v4.50+ (which now uses `\glsuppercase` for all caps commands such as `\GLS`).

`\MFUaddmap`{ ⟨*cs1*⟩ } { ⟨*cs2*⟩ } mfirstuc v2.08+

Identifies a mapping from the command ⟨*cs1*⟩ to command ⟨*cs2*⟩ for `\makefirstuc` and also identifies ⟨*cs2*⟩ as a blocker. Mappings and blockers aren't supported by `\MFUsentencecase`, so both ⟨*cs1*⟩ and ⟨*cs2*⟩ are identified as exclusions for `\MFUsentencecase`.

`\MFUblocker`{ ⟨*cs*⟩ } mfirstuc v2.08+

Locally identifies ⟨*cs*⟩ as a blocker command for `\makefirstuc` and an exclusion for `\MFUsentencecase` (which doesn't support blockers).

`\MFUexcl`{ ⟨*cs*⟩ } mfirstuc v2.08+

Locally identifies ⟨*cs*⟩ as an exclusion command, which will be recognised by both `\makefirstuc` and `\MFUsentencecase`.

`\MFUsave` mfirstuc v2.08+

Saves the list of exclusions, blockers and mappings to the `aux` file (if required by some external tool, such as `bib2gls`). This command sets itself to `\relax` so it doesn't repeat the action if used multiple times, and counteracts any use of `\MFUsaveatend`.

`\MFUsaveatend` mfirstuc v2.08+

Saves the list of exclusions, blockers and mappings to the `aux` file (if required by some external tool, such as `bib2gls`) at the end of the document. This command sets itself to `\relax` so it doesn't repeat the action if used multiple times, but it can be overridden by `\MFUsave`.

`\MFUsentencecase`{ ⟨*text*⟩ } mfirstuc v2.08+

Fully expands ⟨*text*⟩ and converts the first letter to uppercase. Unlike `\makefirstuc`, this command is expandable, but only recognises commands identified as exclusions. See the mfirstuc

documentation for further details. This command is provided by glossaries−extra v1.49+ if an old version of mfirstuc is detected.

\MGP §11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\MGP`. Note that this isn't the same as `\glscapturedgroup`.

\MGLS [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As `\mgls` but uses `\GLS` for each element.

\MGls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As `\mgls` but uses `\Gls` for all elements.

\Mgls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As `\mgls` but uses `\Gls` for the first element.

\mgls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7; 333

References a multi-entry identified by the given ⟨*multi-label*⟩.

\mglsAddOptions { ⟨*multi-label*⟩ } { ⟨*new-options*⟩ } glossaries−extra v1.48+ §7; 335

Locally appends to the options associated with the given multi-entry.

\mglscurrentcategory glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the multi-entry category current in effect.

\mglscurrentlabel glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the current element label.

\mglscurrentlist glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the complete comma-separated list of elements.

\mglscurrentmainlabel glossaries−extra v1.48+ §7.5; 357

Placeholder command for use in multi-entry hooks, this expands to the label of the main element.

\mglscurrentmultilabel glossaries−extra v1.48+ §7.5; 357

Placeholder command for use in multi-entry hooks, this expands to the multi-entry label.

\mglscurrentoptions glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the options used when the multi-entry was defined.

\mglscurrentprefix glossaries−extra v1.48+ §7.5; 358

Placeholder command for use in multi-entry hooks, this expands to the current prefix.

\mglscurrentsuffix glossaries−extra v1.48+ §7.5; 359

Placeholder command for use in multi-entry hooks, this expands to the current suffix.

\mglscustompostlinkhook glossaries−extra v1.48+ §7.6; 359

Hook used with the `mpostlinkelement=custom` option.

\mglsdefcategoryprefix{ ⟨*category-label*⟩ } { ⟨*definition*⟩ }
glossaries−extra v1.48+ §7.3; 352

Defines the prefix for the given multi-entry category.

\mglsdefcategorysuffix{ ⟨*category-label*⟩ } { ⟨*definition*⟩ }
glossaries−extra v1.48+ §7.3; 352

Defines the suffix for the given multi-entry category.

\mglselementindex glossaries−extra v1.48+ §7.5; 358

A count register used in multi-entry hooks, this is set to the element index.

\mglselementposthook glossaries−extra v1.48+ §7.5; 357

Hook performed after each (non-skipped) element in a multi-entry set.

\mglselementprehook glossaries−extra v1.48+ §7.5; 357

Hook performed before each (non-skipped) element in a multi-entry set.

\mglselementreset{ ⟨*entry-label*⟩ } glossaries−extra v1.48+ §7.10; 371

Used by options such as `resetall` to reset an element's first use flag (taking the `preset-local` option into account).

\mglselementunset{ ⟨*entry-label*⟩ } glossaries−extra v1.48+ §7.10; 371

Used by options such as `unsetall` to unset an element's first use flag (taking the `preset-local` option into account).

\mglsfield *initial:* `useri` glossaries−extra v1.48+ §7.11.3; 374

Expands to the internal field label required by `\mglsusefield`.

\mglsforelements { ⟨*multi-label*⟩ } { ⟨*cs*⟩ } { ⟨*body*⟩ } glossaries−extra v1.48+ §7.13; 379

Iterates over the list of element labels for the multi-entry identified by ⟨*multi-label*⟩.

\mglsforotherelements { ⟨*multi-label*⟩ } { ⟨*cs*⟩ } { ⟨*body*⟩ }
glossaries−extra v1.48+ §7.13; 379

As `\mglsforelements` but skips the main entry label.

\Mglsfull [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As `\mglsfull` but sentence case.

\mglsfull [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As `\mgls` but uses `\glsxtrfull` for any elements that have the `short` field set and `\glsfirst` otherwise.

\mglshascategoryprefix { ⟨*category-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.48+ §7.3; 352

Does ⟨*true*⟩ if the given multi-entry category has a prefix set otherwise does ⟨*false*⟩.

\mglshascategorysuffix { ⟨*category-label*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ }
glossaries−extra v1.48+ §7.3; 352

Does ⟨*true*⟩ if the given multi-entry category has a suffix set otherwise does ⟨*false*⟩.

\mglsiflast { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+ §7.5; 359

For use in multi-entry hooks, this expands to ⟨*true*⟩ if this is the last iteration otherwise expands to ⟨*false*⟩.

\mglsiflastelementcapscase{ ⟨*no-change*⟩ } { ⟨*firstuc*⟩ } { ⟨*all caps*⟩ }
glossaries−extra v1.48+

§7.6.1; 361

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*no-change*⟩ if the last element had no case-change applied, to ⟨*firstuc*⟩ if the last element had sentence case applied or to ⟨*all caps*⟩ if the last element had all caps applied.

\mglsiflastelementskipped{ ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+

§7.6.1; 360

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the last element was skipped, otherwise to ⟨*false*⟩.

\mglsiflastelementwasfirstuse{ ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+

§7.6.1; 360

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the last element was used for the first time, otherwise to ⟨*false*⟩.

\mglsiflastelementwasplural{ ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+

§7.6.1; 360

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the last element had the plural form displayed, otherwise to ⟨*false*⟩.

\mglsiflastmaincapscase{ ⟨*no-change*⟩ } { ⟨*firstuc*⟩ } { ⟨*all caps*⟩ }
glossaries−extra v1.48+

§7.6.2; 362

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*no-change*⟩ if the main element from the multi-entry just referenced had no case-change applied, to ⟨*firstuc*⟩ if the last element had sentence case applied or to ⟨*all caps*⟩ if the last element had all caps applied.

\mglsiflastmainskipped{ ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+

§7.6.2; 361

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the main element from the multi-entry just referenced was skipped, otherwise to ⟨*false*⟩.

\mglsiflastmainwasfirstuse{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.48+ §7.6.2; 361

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the main element from the multi-entry just referenced was used for the first time, otherwise to ⟨*false*⟩.

\mglsiflastmainwasplural{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.48+ §7.6.2; 361

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if the main element from the multi-entry just referenced had the plural form shown, otherwise to ⟨*false*⟩.

\mglsisfirstuse{⟨*true*⟩}{⟨*false*⟩} glossaries−extra v1.48+ §7.5; 358

For use in multi-entry hooks, this expands to ⟨*true*⟩ if this is the first use otherwise expands to ⟨*false*⟩.

\mglslastcategory glossaries−extra v1.48+ §7.6; 360

For use in multi-entry suffix and post-link hooks, this expands to the multi-entry category or nothing, if no category assigned.

\mglslastelementlabel glossaries−extra v1.48+ §7.6.1; 360

For use in multi-entry suffix and post-link hooks, this expands to the label of the last non-skipped element.

\mglslastelementpostlinkhook glossaries−extra v1.48+ §7.6; 359

Hook used with the `mpostlinkelement=last` option.

\mglslastmainlabel glossaries−extra v1.48+ §7.6.2; 361

For use in multi-entry suffix and post-link hooks, this expands to the label of the main element that was just referenced.

\mglslastmainpostlinkhook glossaries−extra v1.48+ §7.6; 359

Hook used with the mpostlinkelement=main option.

\mglslastmultilabel glossaries−extra v1.48+ §7.6; 360

For use in multi-entry suffix and post-link hooks, this expands to the multi-entry label.

\mglslocalreset { ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.7; 362

Locally resets the first use flag for the given multi-entry.

\mglslocalunset { ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.7; 362

Locally unsets the first use flag for the given multi-entry.

\mglslocalunsetothers { ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.10; 371

Locally unsets the first use flag for the other (not main) elements of the given multi-entry.

\Mglslong [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As \mglslong but sentence case.

\mglslong [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As \mgls but uses \glsxtrlong for any elements that have the long field set and \glstext otherwise.

\MGLSmainpl [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 373

As \mgls but uses \GLSpl for the main element and \GLS for the others.

\MGlsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.1; 372

As \mgls but uses \Glspl for the main entry and \Gls for the others.

\Mglsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.1; 372

As \mgls uses sentence case for the first element and the plural form for the main element.

\mglsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.1; 372

As \mgls but uses the plural form for the main element.

\MGlsname [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.3; 374

As \mgls but uses \Glsname.

\Mglsname [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.3; 374

As \mgls but uses \Glsname for the first entry and \glsname for the remaining entries.

\mglsname [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.3; 374

As \mgls but uses \glsname.

\MGLSpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries-extra v1.48+

§7.11.1; 373

As \mgls but uses \GLSpl for each element.

\MGlspl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As \mgls but uses \Glspl for each element.

\Mglspl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As \mgls but uses \Glspl for the first element and \glspl for the remaining elements.

\mglspl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.1; 372

As \mgls but uses the plural form for each element.

\mglsprefix glossaries−extra v1.48+ §7.3; 351

Code used to typeset the multi-entry prefix.

\mglsreset {⟨*multi-label*⟩} glossaries−extra v1.48+ §7.7; 362

Globally resets the first use flag for the given multi-entry.

\mglsresetall glossaries−extra v1.48+ §7.7; 363

Resets the first use flag for all multi-entries.

\mglsseefirstitem {⟨*multi-label*⟩} glossaries−extra v1.48+ §7.12; 378

Formatting command used by cross-reference lists for the first item if the item is a multi-entry.

\mglsseeitem {⟨*multi-label*⟩} glossaries−extra v1.48+ §7.12; 378

Formatting command used by cross-reference lists for subsequent items if the item is a multi-entry.

\mglsSetMain { ⟨*multi-label*⟩ } { ⟨*new-main-label*⟩ } glossaries−extra v1.48+

Locally changes the main element for the given multi-entry.

\mglsSetOptions { ⟨*multi-label*⟩ } { ⟨*new-options*⟩ } glossaries−extra v1.48+ §7; 335

Locally sets the options associated with the given multi-entry.

\Mglsshort [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As \mglsshort but sentence case.

\mglsshort [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.2; 373

As \mgls but uses \glsxtrshort for any elements that have the short field set and \glstext otherwise.

\mglssuffix glossaries−extra v1.48+ §7.3; 351

Code used to typeset the multi-entry suffix.

\MGlssymbol [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.3; 374

As \mgls but uses \glssymbol if the symbol field is set and \Gls otherwise.

\Mglssymbol [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.3; 374

As \mgls but uses \glssymbol if the symbol field is set, otherwise it uses \Gls for the first element and \gls for the remaining elements.

\mglssymbol [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.48+ §7.11.3; 374

As \mgls but uses \glssymbol if the symbol field is set and \gls otherwise.

\mglsunset { ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.7; 362

Globally unsets the first use flag for the given multi-entry.

\mglsunsetall glossaries−extra v1.48+ §7.7; 362

Unsets the first use flag for all multi-entries.

\mglsunsetothers { ⟨*multi-label*⟩ } glossaries−extra v1.48+ §7.10; 371

Globally unsets the first use flag for the other (not main) elements of the given multi-entry.

\mglsusecategoryprefix { ⟨*category-label*⟩ } glossaries−extra v1.48+ §7.3; 352

Expands to the prefix assigned to the given multi-entry category or does nothing if no prefix assigned.

\mglsusecategorysuffix { ⟨*category-label*⟩ } glossaries−extra v1.48+ §7.3; 352

Expands to the suffix assigned to the given multi-entry category or does nothing if no suffix assigned.

\MGlsusefield [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.3; 375

As \mglsusefield but sentence case for each element.

\Mglsusefield [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.3; 375

As \mglsusefield but sentence case for the first element.

\mglsusefield [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.3; 374

As \mgls but uses \glsdisp if the field identified by \mglsfield exists with the link text obtained from the field value.

\mglswasfirstuse { ⟨*true*⟩ } { ⟨*false*⟩ } glossaries−extra v1.48+ §7.6; 360

For use in multi-entry suffix and post-link hooks, this expands to the ⟨*true*⟩ if this was the first use of the multi-entry, otherwise to ⟨*false*⟩.

\MPGLS [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.4; 377

As \mpgls but all caps for the all elements.

\MPGls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.4; 377

As \mpgls but sentence case for all elements.

\Mpgls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.4; 376

As \mpgls but sentence case for the first element.

\mpgls [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.48+ §7.11.4; 376

As \mgls but uses \pgls for the first element.

\MPGLSmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 377

As \mpglsmainpl but all caps for the all elements.

\MPGlsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 377

As \mpglsmainpl but sentence case for all elements.

\Mpglsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 377

As \mpglspl but sentence case for the first element.

\mpglsmainpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 376

As \mgls but uses \pglspl for the first element if its the main element otherwise \pgls and, for the remaining elements, uses \glspl if the element is the main entry or \gls otherwise.

\MPGLSpl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 377

As \mpglspl but all caps for the all elements.

\MPGlspl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 377

As \mpglspl but sentence case for all elements.

\mpglspl [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries-extra v1.48+ §7.11.4; 376

As \mgls but uses \pglspl for the first element and \glspl for the remaining elements.

\mpglsWarning glossaries−extra v1.48+ §7.11.4; 376

Issues a warning with \GlossariesExtraWarning indicating that glossaries−prefix is required for \mpgls family of commands.

\Mu glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{M}.

\multiglossaryentry [⟨*options*⟩] { ⟨*multi-label*⟩ } [⟨*main-label*⟩] { ⟨*entry-label-list*⟩ } glossaries−extra v1.48+ §7; 333

Defines a multi-entry set with the label ⟨*multi-label*⟩, consisting of the entries whose labels are listed in ⟨*entry-label-list*⟩, where the main entry (which must be present in ⟨*entry-label-list*⟩) is identified by ⟨*main-label*⟩ (or the final element in ⟨*entry-label-list*⟩, if ⟨*main-label*⟩ is omitted).

\multiglossaryentryglobalfalse glossaries−extra v1.48+ §7; 335

Sets \ifmultiglossaryentryglobal to false.

\multiglossaryentryglobaltrue glossaries−extra v1.48+ §7; 334

Sets \ifmultiglossaryentryglobal to true.

\multiglossaryentrysetup { ⟨*options*⟩ } glossaries−extra v1.48+ §7.9; 364

Specifies a general set of options to apply to all multi-entries.

N

\newabbr [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*short*⟩ } { ⟨*long*⟩ } §4.3.2; Table 4.1

A synonym for \newabbreviation defined by the shortcuts=abbreviations or shortcuts=ac package option.

\newabbreviation [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*short*⟩ } { ⟨*long*⟩ }

§4.1; 42

Defines a new entry that represents an abbreviation. This internally uses `\newglossaryentry` and any provided ⟨*options*⟩ (glossary entry keys) will be appended. The `category` is set to `abbreviation` by default, but may be overridden in ⟨*options*⟩. The appropriate style should be set before the abbreviation is defined with `\setabbreviationstyle`.

\newabbreviationhook

§4.1.5; 46

Hook provided within `\newabbreviation` just before the entry is defined.

\newabbreviationstyle { ⟨*style-name*⟩ } { ⟨*setup*⟩ } { ⟨*display definitions*⟩ }

§4.5.3; 166

Defines an abbreviation style, which can be set with `\setabbreviationstyle`.

\newacronym [⟨*options*⟩] { ⟨*entry-label*⟩ } { ⟨*short*⟩ } { ⟨*long*⟩ } glossaries

This command is provided by the base glossaries package but is redefined by glossaries−extra to use `\newabbreviation` with the `category` key set to `acronym`. The appropriate style should be set before the abbreviation is defined with `\setabbreviationstyle [acronym] {`⟨*style*⟩`}`. You can override the `category` in ⟨*options*⟩ but remember to change the optional argument of `\setabbreviationstyle` to match.

⊘ **\newacronymstyle** { ⟨*name*⟩ } { ⟨*format def*⟩ } { ⟨*display defs*⟩ } glossaries v4.02+

Defines an acronym style for use with the base glossaries package's acronym mechanism. These styles are not compatible with glossaries−extra. Use `\newabbreviationstyle` instead.

\newdglsfield [⟨*default-options*⟩] { ⟨*field*⟩ } { ⟨*cs*⟩ } glossaries−extra−bib2gls v1.49+

§11.6.7; 615

Defines the command ⟨*cs*⟩[⟨*options*⟩]{⟨*entry-label*⟩} to behave like `\dglsfield[`⟨*default-options*⟩`,`⟨*options*⟩`]{`⟨*entry-label*⟩`}{`⟨*field*⟩`}`.

\newdglsfieldlike [⟨*default-options*⟩] {⟨*field*⟩} {⟨*cs*⟩} {⟨*Cs*⟩} {⟨*CS*⟩}
glossaries-extra-bib2gls v1.49+

§11.6.7; 615

Similar to `\newdglsfield` but also defines sentence case (⟨*Cs*⟩) and all caps (⟨*CS*⟩) commands with mappings.

\newentry {⟨*entry-label*⟩} {⟨*options*⟩}

§2.4; 19

A synonym for `\newglossaryentry` defined by the `shortcuts=other` package option.

\newglossary [⟨*log-ext*⟩] {⟨*glossary-label*⟩} {⟨*in-ext*⟩} {⟨*out-ext*⟩} {⟨*title*⟩} [⟨*counter*⟩] glossaries

Defines a glossary identified by ⟨*glossary-label*⟩ (which can be referenced by the `type` key when defining an entry). The ⟨*title*⟩ will be used when displaying the glossary (using commands like `\printglossary`), but this title can be overridden by the `title` option. The optional ⟨*counter*⟩ indicates which counter should be used by default for the location when indexing any entries that have been assigned to this glossary. (This can be overridden by the `counter` option.) The other arguments are file extensions for use with `makeindex` or `xindy`. These arguments aren't relevant for other indexing options (in which case, you may prefer to use `\newglossary*`).

\newglossary* {⟨*glossary-label*⟩} {⟨*title*⟩} [⟨*counter*⟩] glossaries v4.08+

A shortcut that supplies file extensions based on the glossary label:

`\newglossary`[⟨*glossary-label*⟩`-glg`]{⟨*glossary-label*⟩}{⟨*glossary-label*⟩`-gls`}{⟨⟨*glossary-label*⟩`-glo`⟩}{⟨*title*⟩}[⟨*counter*⟩]

\newglossaryentry {⟨*entry-label*⟩} {⟨*key=value list*⟩} glossaries

Defines a new glossary entry with the given label. The second argument is a comma-separated list of glossary entry keys.

\newignoredglossary { ⟨*glossary-label*⟩ } glossaries v4.08+

Defines a glossary that should be ignored by iterative commands, such as `\printglossaries`. This glossary has no associated indexing files and has hyperlinks disabled. You can use an ignored glossary for common terms or abbreviations that don't need to be included in any listing (but you may want these terms defined as entries to allow automated formatting with the `\gls`-like commands). An ignored glossary can't be displayed with `\printglossary` but may be displayed with the "unsrt" family of commands, such as `\printunsrtglossary`.

\newignoredglossary* { ⟨*glossary-label*⟩ } glossaries−extra v1.11+ §8; 384

This is like the unstarred `\newignoredglossary` but doesn't disable hyperlinks. You will need to ensure that the hypertargets are defined. For example, with `\printunsrtglossary` or through standalone entrys.

\newnum [⟨*key=value list*⟩] { ⟨*entry-label*⟩ } { ⟨*num*⟩ } §2.4; 20

A synonym for `\glsxtrnewnumber` defined by the `shortcuts=other` package option (provided the `numbers` option is also used).

\newsym [⟨*key=value list*⟩] { ⟨*entry-label*⟩ } { ⟨*sym*⟩ } §2.4; 19

A synonym for `\glsxtrnewsymbol` defined by the `shortcuts=other` package option (provided the `symbols` option is also used).

\newterm [⟨*key=value list*⟩] { ⟨*entry-label*⟩ } glossaries v4.02+ §2.1; 13
(requires `index` package option)

Defines a new glossary entry with the given label, `type` set to `index`, the `name` set to ⟨*entry-label*⟩ and the `description` set to `\nopostdesc`. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.

\NIN §11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\NIN`.

\nopostdesc glossaries v1.17+

When placed at the end of the description, this switches off the post-description hook (including the post-description punctuation). Does nothing outside of the glossary.

\NOTPREFIXOF §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NOTPREFIXOF.

\NOTSUFFIXOF §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NOTSUFFIXOF.

\Nu glossaries-extra-bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{N}.

\NULL §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NULL.

O

\Omicron glossaries-extra-bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{O}.

\omicron glossaries-extra-bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{O}.

P

\pagelistname *initial:* `Page List` glossaries
(language-sensitive)

Expands to the title of the location list column for headed tabular-like styles.

\PGLS [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

As `\pgls` but all caps.

\Pgls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

As `\pgls` but sentence case.

\pgls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

Similar to `\gls` but inserts the appropriate prefix, if provided.

\PGLSfmtlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.2; 213

As `\pglsfmtlong` but all caps.

\Pglsfmtlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.2; 213

As `\pglsfmtlong` but sentence case.

\pglsfmtlong [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+

§5.3.2; 213

As `\glsfmtlong` but inserts the `prefixfirst` field and separator in front if set.

\PGLSfmtlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 214

As \pglsfmtlongpl but all caps.

\Pglsfmtlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 214

As \pglsfmtlongpl but sentence case.

\pglsfmtlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 213

As \glsfmtlongpl but inserts the prefixfirstplural field and separator in front if set.

\PGLSfmtshort [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 211

As \pglsfmtshort but all caps.

\Pglsfmtshort [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 211

As \pglsfmtshort but sentence case.

\pglsfmtshort [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§5.3.2; 211

As \glsfmtshort but inserts the prefix field and separator in front if set.

\PGLSfmtshortpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.2; 212

As \pglsfmtshortpl but all caps.

\Pglsfmtshortpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.2; 212

As \pglsfmtshortpl but sentence case.

\pglsfmtshortpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.2; 212

As \glsfmtshortpl but inserts the prefixplural field and separator in front if set.

\PGLSpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

As \pgls but all caps.

\Pglspl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

As \pgls but sentence case.

\pglspl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−prefix

Similar to \glspl but inserts the appropriate prefix, if provided.

\PGLSxtrlong [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 57

As \pglsxtrlong but all caps.

\Pglsxtrlong [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 56

As \pglsxtrlong but sentence case.

\pglsxtrlong [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 56

As \glsxtrlong but inserts the prefixfirst field and separator in front if set.

\PGLSxtrlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 57

As \pglsxtrlongpl but all caps.

\Pglsxtrlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 57

As \pglsxtrlongpl but sentence case.

\pglsxtrlongpl [⟨*options*⟩] {⟨*entry-label*⟩} [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries-extra v1.49+
(requires glossaries-prefix)

§4.3.1; 57

As \glsxtrlongpl but inserts the prefixfirstplural field and separator in front if set.

\PGLSxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \pglsxtrshort but all caps.

\Pglsxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \pglsxtrshort but sentence case.

\pglsxtrshort [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \glsxtrshort but inserts the prefix field and separator in front if set.

\PGLSxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \pglsxtrshortpl but all caps.

\Pglsxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \pglsxtrshortpl but sentence case.

\pglsxtrshortpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * +
⟨*alt-mod*⟩ glossaries−extra v1.49+
(requires glossaries−prefix)

§4.3.1; 56

As \glsxtrshortpl but inserts the prefixplural field and separator in front if set.

\Pglsxtrtitlelong{⟨*entry-label*⟩} glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.3; 223

The normal behaviour of `\Pglsfmtlong`.

\Pglsxtrtitlelongpl{⟨*entry-label*⟩} glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.3; 223

The normal behaviour of `\Pglsfmtlongpl`.

\Pglsxtrtitleshort{⟨*entry-label*⟩} glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.3; 223

The normal behaviour of `\Pglsfmtshort`.

\Pglsxtrtitleshortpl{⟨*entry-label*⟩} glossaries−extra v1.49+
(requires glossaries−prefix)

§5.3.3; 223

The normal behaviour of `\Pglsfmtshortpl`.

\predglsfieldhook{⟨*entry-label*⟩} glossaries−extra−bib2gls v1.8+

§11.6.7; 616

Hook used by `\dglsfield`-like commands.

\predglshook{⟨*entry-label*⟩} glossaries−extra−bib2gls v1.8+

§11.6.7; 616

Hook used by `\dgls`-like commands.

\predglslinkhook{⟨*entry-label*⟩} glossaries−extra−bib2gls v1.8+

§11.6.7; 616

Hook used by `\dglslink` and `\dglsdisp`.

\PREFIXOF

§11.6.2; 582

Defined by `\GlsXtrResourceInitEscSequences` to expand to detokenized `\PREFIXOF`.

\pretoglossarypreamble[⟨*type*⟩]{⟨*text*⟩} glossaries−extra v1.12+ §8.2; 386

Prepends (locally) ⟨*text*⟩ to the preamble for the glossary identified by ⟨*type*⟩. If ⟨*type*⟩ is omitted, `\glsdefaulttype` is assumed.

\printabbreviations[⟨*options*⟩] (requires `\usepackage[abbreviations]{glossaries-extra}`) §2.1; 10

Shortcut for `\printglossary[type=\glsxtrabbrvtype]`.

\printacronyms[⟨*options*⟩] glossaries
(requires the `acronyms` package option)

Shortcut for `\printglossary[type=\acronymtype]`.

\printglossaries glossaries

Iterates over all non-ignored glossaries and does `\printglossary[type=`⟨*type*⟩`]` for each glossary.

\printglossary[⟨*options*⟩] glossaries

Displays the glossary by inputting a file created by `makeindex` or `xindy`. Must be used with `\makeglossaries` and either `makeindex` or `xindy`.

\printindex[⟨*options*⟩] `v4.02+` (requires the `index` package option)

Shortcut provided by the `index` package option that simply does `\printglossary[type=index]`.

\printnoidxglossaries glossaries v4.04+

Iterates over all non-ignored glossaries and does `\printnoidxglossary[type=`⟨*type*⟩`]` for each glossary.

\printnoidxglossary [⟨*options*⟩] glossaries v4.04+

Displays the glossary by obtaining the indexing information from the `aux` file and using TEX to sort and collate. Must be used with `\makenoidxglossaries` or with the glossaries not identified in the optional argument of `\makeglossaries` when using the hybrid method. This method can be very slow and has limitations.

\printnumbers [⟨*options*⟩] glossaries v4.02+
(requires the `numbers` package option)

Shortcut for `\printglossary[type=numbers]`.

\printsymbols [⟨*options*⟩] glossaries v4.02+
(requires the `symbols` package option)

Shortcut for `\printglossary[type=symbols]`.

\printunsrtabbreviations [⟨*options*⟩] glossaries-extra-bib2gls v1.40+ §11.6.1; 580
(requires `\usepackage[abbreviations,record]{glossaries-extra}`)

Shortcut for `\printunsrtglossary[type=\glsxtrabbrvtype]`.

\printunsrtacronyms [⟨*options*⟩] glossaries-extra-bib2gls v1.40+ §11.6.1; 580
(requires `\usepackage[acronyms,record]{glossaries-extra}`)

Shortcut for `\printunsrtglossary[type=\acronymtype]`.

\printunsrtglossaries glossaries-extra v1.08+ §8.4; 393

Iterates over all non-ignored glossaries and does `\printunsrtglossary[type=`⟨*type*⟩`]` for each glossary.

\printunsrtglossary [⟨*options*⟩] glossaries-extra v1.08+ §8.4; 392

Displays the glossary by iterating over all entries associated with the given glossary (in the order in which they were added to the glossary). Group headers will only be inserted if the `group`

key has been defined and has been set (typically with the `record` option and `bib2gls`). Location lists will only be shown if the `location` or `loclist` fields have been set (typically by `bib2gls`).

\printunsrtglossary* [⟨*options*⟩] { ⟨*init-code*⟩ } glossaries−extra v1.12+ §8.4; 392

Does { ⟨*init-code*⟩ `\printunsrtglossary` [⟨*options*⟩] } which localises ⟨*init-code*⟩.

\printunsrtglossaryentryprocesshook { ⟨*entry-label*⟩ } glossaries−extra v1.21+ §8.4.3; 414

Hook used within the "unsrt" family of commands while the glossary is being constructed.

\printunsrtglossarygrouphook { ⟨*internal cs*⟩ } glossaries−extra v1.50+ §8.4.1; 403

Hook used within the "unsrt" family of commands while the group header is being constructed.

\printunsrtglossaryhandler { ⟨*entry-label*⟩ } glossaries−extra v1.16+ §8.4.3; 415

Used within the "unsrt" family of commands to process the current entry.

\printunsrtglossarypostbegin { ⟨*internal cs*⟩ } glossaries−extra v1.50+ §8.4.3; 412

Hook used within the "unsrt" family of commands while the glossary is being constructed just after `\begin{theglossary}` is added.

\printunsrtglossarypostentryprocesshook { ⟨*internal cs*⟩ } glossaries−extra v1.50+ §8.4.3; 415

Hook used within the "unsrt" family of commands while the glossary is being constructed after the entry line has been added.

\printunsrtglossarypredoglossary glossaries−extra v1.21+ §8.4.3; 415

Hook performed by the "unsrt" family of commands just before the glossary body is displayed.

\printunsrtglossarypreend{⟨*internal cs*⟩} glossaries−extra v1.50+ §8.4.3; 413

Hook used within the "unsrt" family of commands while the glossary is being constructed just before `\end{theglossary}` is added.

\printunsrtglossarypreentryprocesshook{⟨*internal cs*⟩}
glossaries−extra v1.50+ §8.4.3; 414

Hook used within the "unsrt" family of commands while the glossary is being constructed before the entry line has been added.

\printunsrtglossaryskipentry glossaries−extra v1.21+ §8.4.3; 414

May be used within `\printunsrtglossaryentryprocesshook` to skip the current entry.

\printunsrtglossaryunit[⟨*options*⟩]{⟨*counter-name*⟩}
glossaries−extra v1.12+ §8.4.3.2; 422

Provided for use with `\GlsXtrRecordCounter` to display a glossary with `\printunsrtglossary*` that filters entries that don't have a match for the current ⟨*counter-name*⟩ value.

\printunsrtglossaryunitpostskip glossaries−extra v1.49+ §8.4.3.2; 423

The vertical space at the end of the glossary appended by `\printunsrtglossaryunit`.

\printunsrtglossaryunitsetup{⟨*counter-name*⟩} glossaries−extra v1.12+ §8.4.3.2; 423

Sets up the filtering used by `\printunsrtglossaryunit`.

\printunsrtindex[⟨*options*⟩] glossaries−extra−bib2gls v1.40+ §11.6.1; 581
(requires `\usepackage[index,record]{glossaries-extra}`)

Shortcut provided by the `index` package option combined with glossaries−extra−bib2gls that simply does `\printunsrtglossary[type=index]`.

\printunsrtinnerglossary [⟨*options*⟩] {⟨*pre-code*⟩} {⟨*post-code*⟩}
glossaries−extra v1.44+

§8.4.3.1; 417

Similar to \printunsrtglossary but doesn't contain the code that starts and ends the glossary (such as beginning and ending the theglossary environment), so this command needs to be either placed inside printunsrtglossarywrap or in the \printunsrtglossary entry handler \printunsrtglossaryhandler.

\printunsrtnumbers [⟨*options*⟩] glossaries−extra−bib2gls v1.40+
(requires \usepackage[numbers,record]{glossaries-extra})

§11.6.1; 581

Shortcut provided by the numbers package option combined with glossaries−extra−bib2gls that simply does \printunsrtglossary[type=numbers].

\printunsrtsymbols [⟨*options*⟩] glossaries−extra−bib2gls v1.40+
(requires \usepackage[symbols,record]{glossaries-extra})

§11.6.1; 581

Shortcut for \printunsrtglossary[type=symbols].

\printunsrttable [⟨*options*⟩] glossary−table v1.49+

§8.7.4; 491

Internally uses \printunsrtglossary with the table style.

\provideignoredglossary {⟨*glossary-label*⟩} *modifier:* *
glossaries−extra v1.12+

§8; 384

As \newignoredglossary but does nothing if the glossary has already been defined.

\providemultiglossaryentry [⟨*options*⟩] {⟨*multi-label*⟩} [⟨*main-label*⟩]
{⟨*entry-label-list*⟩} glossaries−extra v1.48+

§7; 334

As \multiglossaryentry but does nothing if a multi-entry set has already been defined with the given label.

\ProvidesGlossariesExtraLang{ ⟨*tag*⟩ }

§15; 643

Should be placed at the start of a glossaries−extra `ldf` file.

R

\renewabbreviationstyle{ ⟨*style-name*⟩ } { ⟨*setup*⟩ } { ⟨*display definitions*⟩ }
glossaries−extra v1.04+

§4.5.3; 166

Redefines an abbreviation style.

\RequireGlossariesExtraLang{ ⟨*tag*⟩ }

Indicates that a glossaries−extra `ldf` file should be input, if it hasn't already been input.

\RestoreAcronyms

§4.6; 186

Restores `\newacronym` to the base glossaries mechanism. Not recommended.

\rGLS [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+

§11.5; 572

Like `\GLS` but hooks into the entry's record count.

\rGls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+

§11.5; 572

Like `\Gls` but hooks into the entry's record count.

\rgls [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+

§11.5; 571

Like `\gls` but hooks into the entry's record count.

\rGLSformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 572

Format used by \rGLS if the entry's record count is more than the given trigger value.

\rGlsformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 572

Format used by \rGls if the entry's record count is more than the given trigger value.

\rglsformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 571

Format used by \rgls if the entry's record count is more than the given trigger value.

\rGLSpl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+ §11.5; 572

Like \GLSpl but hooks into the entry's record count.

\rGlspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+ §11.5; 572

Like \Glspl but hooks into the entry's record count.

\rglspl [⟨*options*⟩] { ⟨*entry-label*⟩ } [⟨*insert*⟩] *modifiers:* * + ⟨*alt-mod*⟩
glossaries−extra v1.21+ §11.5; 571

Like \glspl but hooks into the entry's record count.

\rGLSplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 572

Format used by \rGLSpl if the entry's record count is more than the given trigger value.

\rGlsplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 572

Format used by \rGlspl if the entry's record count is more than the given trigger value.

\rglsplformat { ⟨*entry-label*⟩ } { ⟨*insert*⟩ } glossaries−extra v1.21+ §11.5; 572

Format used by `\rglspl` if the entry's record count is more than the given trigger value.

\Rho glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with `\providecommand`, this just does `\mathrm{P}`.

S

\seealsoname *initial:* `see also` glossaries−extra v1.16+
(language-sensitive)

Used as a cross-reference tag. The default value is `\alsoname`, if that command has been defined, or "see also".

\seename *initial:* `see` (language-sensitive)

Used as a cross-reference tag (provided by glossaries if not already defined).

\setabbreviationstyle [⟨*category*⟩] { ⟨*style-name*⟩ } §4.5; 60

Sets the current abbreviation style to ⟨*style-name*⟩ for the category identified by ⟨*category*⟩. If the optional argument is omitted, `abbreviation` is assumed.

⊘ **\setacronymstyle** { ⟨*acronym-style-name*⟩ } glossaries v4.02+

Sets the style for the base glossaries package's acronym mechanism. These styles are not compatible with glossaries−extra, which redefines `\newacronym` to use `\newabbreviation`. Use:

```
\setabbreviationstyle[acronym]{⟨abbreviation-style-name⟩}
```

with the closest matching abbreviation style instead.

\setentrycounter [⟨*prefix*⟩] { ⟨*counter*⟩ } glossaries

Used to set the location counter and prefix required for \glshypernumber.

\setglossarypreamble [⟨*type*⟩] { ⟨*text*⟩ } glossaries v3.07+

Globally sets the preamble for the glossary identified by ⟨*type*⟩ to ⟨*text*⟩. If ⟨*type*⟩ is omitted, \glsdefaulttype is assumed.

\setglossarystyle { ⟨*style-name*⟩ } glossaries v3.08a+

Set the current glossary style to ⟨*style-name*⟩. Redefined by glossaries−extra to include \glsxtrpreglossarystyle.

\setupglossaries { ⟨*options*⟩ } glossaries v3.11a+ §2; 10

Change allowed options that are defined by the base glossaries package. Note that some options can only be passed as package options. To change options defined or modified by the glossaries−extra package, use \glossariesextrasetup.

\setupglsadd { ⟨*options*⟩ } glossaries−extra v1.49+ §5.1.1; 194

Set the default options for \glsadd.

\setupglslink { ⟨*options*⟩ } glossaries−extra v1.49+ §5.1.1; 194

Set the default \glslink options.

\subglossentry { ⟨*level*⟩ } { ⟨*entry label*⟩ } { ⟨*location list*⟩ } glossaries v3.08+

Used to format a child entry. This command should be redefined by the glossary style.

\SUFFIXOF §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \SUFFIXOF.

\symbolname *initial:* Symbol glossaries
(language-sensitive)

Expands to the title of the symbol column for headed tabular-like styles.

T

\Tau glossaries-extra-bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{T}.

\theglossaryentry glossaries v3.0+

The value of the glossaryentry counter.

\TITLE §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \TITLE.

\TRIM §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \TRIM.

U

\u⟨*hex*⟩ §11.6.2; 582

Recognised by bib2gls within some resource options as identifying the Unicode character given by ⟨*hex*⟩. Since \u is defined by the LaTeX kernel as an accent command, you need to protect it from expansion while the options are written to the aux file (\string\u⟨*hex*⟩).

\UC §11.6.2; 582

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \UC.

\Upalpha glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{A}`.

\Upbeta glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{B}`.

\Upchi glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{X}`.

\Upepsilon glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{E}`.

\Upeta glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{H}`.

\Upiota glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{I}`.

\Upkappa glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{K}`.

\Upmu glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{M}`.

\Upnu glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{N}`.

\Upomicron glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{O}`.

\upomicron glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{o}`.

\Uprho glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{P}`.

\Uptau glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{T}`.

\Upzeta glossaries-extra-bib2gls v1.27+ §11.6.8; 623

Defined with `\providecommand` and only if upgreek has been loaded, this just does `\mathrm{Z}`.

W

\writemultiglossentry { ⟨*options*⟩ } { ⟨*multi-label*⟩ } { ⟨*main-label*⟩ } { ⟨*list*⟩ }
glossaries−extra v1.48+ §7.13; 380

Writes multi-entry information to the `aux` file.

X

\xcapitalisefmtwords { ⟨*text*⟩ } mfirstuc v2.03+

Passes the argument to `\capitalisefmtwords` but with the first token in ⟨*text*⟩ expanded. The starred version uses the starred version of `\capitalisefmtwords`.

\xglssetwidest [⟨*level*⟩] { ⟨*name*⟩ } glossaries−extra−stylemods v1.05+ §8.6.5.4; 450

As `\eglssetwidest` but global.

\xglsupdatewidest [⟨*level*⟩] { ⟨*name*⟩ } glossaries−extra−stylemods v1.23+ §8.6.5.4; 450

As `\eglsupdatewidest` but global.

\xGlsXtrIfValueInFieldCsvList { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ }
{ ⟨*value*⟩ } { ⟨*true*⟩ } { ⟨*false*⟩ } *modifier:* * glossaries−extra v1.47+ §5.13; 312

As `\GlsXtrIfValueInFieldCsvList` but fully expands ⟨*value*⟩.

\xGlsXtrSetField { ⟨*entry-label*⟩ } { ⟨*field-label*⟩ } { ⟨*value*⟩ }
glossaries−extra v1.12+ §3.5; 41

As `\GlsXtrSetField` but expands the value and uses a global assignment.

\xpglsxtrpostabbrvfootnote glossaries−extra v1.49+ §4.5.2; 165

Expands to the footnote code required for styles like short−postfootnote.

\xpglsxtrposthyphenlong glossaries−extra v1.49+ 158

Used within the post-link hook for the short−hyphen−postlong−hyphen style on first use. This expands the placeholder commands and uses either \glsxtrposthyphenlong or \GLSxtrposthyphenlong.

\xpglsxtrposthyphenshort glossaries−extra v1.49+ 156

Used within the post-link hook for the long−hyphen−postshort−hyphen style on first use. This expands the placeholder commands and uses either \glsxtrposthyphenshort or \GLSxtrposthyphenshort.

\xpglsxtrposthyphensubsequent glossaries−extra v1.49+ 157

Used within the post-link hook for the long−hyphen−postshort−hyphen style on subsequent use. This expands the placeholder commands and uses either \glsxtrposthyphensubsequent or \GLSxtrposthyphensubsequent.

Z

\Zeta glossaries−extra−bib2gls v1.27+ §11.6.8; 622

Defined with \providecommand, this just does \mathrm{Z}.

Environment Summary

\begin{**glstablesubentries**} glossary−table v1.49+

§8.7.4.1; 497, 818, *993*

Encapsulates the child list.

\begin{**printunsrtglossarywrap**} [⟨*options*⟩] glossaries−extra v1.44+

§8.4.3.1; 386–388, 392, 413, 417, 418, 652, 672, 673, 983, *993*

Sets up the start and end of the glossary (including beginning and ending the theglossary environment). Use \printunsrtinnerglossary within the body for each block of entries.

\begin{**theglossary**} glossaries

3, 411, 412, 417, 418, 723, 804, 983, *993*, 1001

Redefined by glossary styles to set up the way the glossary is displayed. For example, to start and end the description environment for the list styles.

Package Option Summary

`\usepackage[`⟨*options*⟩`]{`**`glossaries-extra-stylemods`**`}` (or `\usepackage[stylemods=`⟨*options*⟩`]{glossaries-extra}`)

Modifies the glossary styles supplied with the base glossaries package to make them more flexible and to integrate support for features provided by glossaries−extra or `bib2gls`.

`all`
Load all predefined styles.

⟨*name*⟩
Load package glossary-⟨*name*⟩.

`\usepackage[`⟨*options*⟩`]{`**`glossaries-extra`**`}`

Extension package that loads glossaries, provides additional commands, and modifies some of the base glossaries commands to integrate them with the new commands or to make them more flexible.

`abbreviations`
Provides a new glossary with the label `abbreviations` and title given by `\abbreviationsname`, redefines `\glsxtrabbrvtype` to `abbreviations`, redefines `\acronymtype` to `\glsxtrabbrvtype` (unless the `acronym` or `acronyms` option has been used), and provides `\printabbreviations`.

`accsupp`
Loads glossaries−accsupp.

`autoseeindex`=⟨*boolean*⟩ *default:* `true`; *initial:* `true`
Indicates whether or not to enable automatic indexing of `see` and `seealso` fields.

`autoseeindex=`**`false`**
Disables automatic indexing of `see` and `seealso` fields.

`autoseeindex=`**`true`**
Enables automatic indexing of `see` and `seealso` fields.

bibglsaux=⟨*basename*⟩ *initial: empty* glossaries−extra v1.49+ (requires bib2gls v3.0+)
§2.4; 27, 709, 917, *995*

If set, an additional aux file called ⟨*basename*⟩.aux will be created in which to store the bib2gls records. This file will be skipped by LaTeX when the main aux file is input but will be read by bib2gls.

debug=⟨*value*⟩ *default:* true; *initial:* false
§2.5; 29, *995*, 1000

Provides debugging information. Some options are also available with just the base glossaries package.

debug=**all** glossaries−extra v1.21+
29, *995*

Implements all debugging options.

debug=**false** glossaries v4.24+
29, *995*

Disable debugging actions.

debug=**showaccsupp** glossaries v4.45+
29. *995* 30, 31, 811, 812, *995*, 998, *see also* showtargets, 30, 946, *995*

Implements debug=true and also shows accessibility information in the document.

debug=**showtargets** glossaries v4.24+

Implements debug=true and also shows target markers in the document.

debug=**showwrgloss** glossaries−extra v1.21+

Implements debug=true and also shows a marker in the document whenever the entry is indexed and (if used with indexcounter) will mark where the wrglossary counter is incremented.

debug=**true** glossaries v4.24+
29, 30, *995*

Writes wrglossary(⟨*type*⟩)(⟨*indexing info*⟩) to the log file if there is an attempt to index an entry before the associated indexing file has been opened (makeindex and xindy only). With glossaries−extra, this setting will also display the label of any undefined entries that are referenced in the document.

docdef=⟨*value*⟩ *default:* true; *initial:* false
§2.4; 17, 334, 378, 380, 628, *995*

Determines whether or not \newglossaryentry is permitted in the document environment.

docdef=**atom**
18, *995*

As restricted but creates the glsdefs file for atom's autocomplete support.

docdef=**false**
17, *995*

Don't allow \newglossaryentry in the document environment.

docdef=**restricted**
18, *995*

Allow \newglossaryentry in the document environment, but only before any glossaries.

`docdef=`**`true`**
Allow `\newglossaryentry` in the document environment if the base glossaries package would allow it.

`equations`=⟨*boolean*⟩ *default:* `true`; *initial:* `false`
Automatically switch the location counter to equation when inside a numbered equation environment.

`floats`=⟨*boolean*⟩ *default:* `true`; *initial:* `false`
Automatically switch the location counter to the corresponding counter when inside a floating environment.

`indexcounter`
Defines the index counter wrglossary and implements `counter=wrglossary`.

`indexcrossrefs`=⟨*boolean*⟩ *default:* `true`; *initial: varies*
Indicates whether or not to enable automatic indexing of cross-referenced entries.

`indexcrossrefs=`**`false`**
Disables automatic indexing of cross-referenced entries.

`indexcrossrefs=`**`true`**
Enables automatic indexing of cross-referenced entries.

`nomissingglstext`=⟨*boolean*⟩ *default:* `true`; *initial:* `false`
Determines whether or not to display warning text if the external glossary file hasn't been generated due to an incomplete build.

`postdot` glossaries-extra v1.12+
A shortcut for `nopostdot=false`.

`postpunc`=⟨*value*⟩ glossaries-extra v1.21+
An alternative to `postdot`, this can be used to insert a different punctuation character after the description.

`postpunc=`**`comma`**
Inserts a comma after the description.

`postpunc=`**`dot`**
Equivalent to `postdot` or `nopostdot=false`.

`postpunc=`**`none`**
Switches off automatic post-description punctuation insertion.

`postpunc=`⟨*punctuation*⟩
Inserts ⟨*punctuation*⟩ after the description.

prefix
Loads glossaries-prefix.

§2.3; 15, 55, 376, *997*

record=⟨*value*⟩ *default:* only; *initial:* off
Indicates whether or not bib2gls is being used (in which case entry indexing is performed by adding bib2gls records in the aux file).

§2.4; 24, *997*

record=**alsoindex** *alias:* hybrid

record=**hybrid**
Performs a mixture of bib2gls records in the aux file (to select entries from a bib file) and makeindex/xindy indexing in their associated files. The actual sorting and collation is performed by the indexing application, so sort=none and save-locations=false should be used in \GlsXtrLoadResources (because it's redundant to make bib2gls sort and collate as well). This setting should be used with \makeglossaries before \GlsXtrLoadResources and glossaries should be displayed with \printglossary (or \printglossaries). There's little point in using this setting unless you have a custom xindy module that you can't convert to an equivalent set of bib2gls options.

see 26, 30, 38, 541, 543, 580, *997*

25, 35, 200, 288, 380, 399, 541, 542, 547, 580, 604, 650, 656, 833, 851, 997

record=**nameref**
Entry indexing is performed by adding bib2gls nameref records in the aux file. Glossaries should be displayed with the "unsrt" family of commands.

record=**off**
Entry indexing is performed as per the base glossaries package, using either \makeglossaries or \makenoidxglossaries.

24, 38, 395, *997*

record=**only**
Entry indexing is performed by adding bib2gls records in the aux file. Glossaries should be displayed with the "unsrt" family of commands.

24, 25, 35, 38, 288, 380, 399, 542, 580, 656, 833, 997

shortcuts={⟨*value*⟩} *initial:* none
Defines various shortcut commands (boolean only with just the base glossaries package).

§2.4; 19, 20, 318, 336, *997*, 998

shortcuts=**abbr** *alias:* abbreviations

shortcuts=**abbreviations**
Implements the abbreviation shortcuts.

Table 4.1; 19, 20, 57, 699, 700, 706–709, 849, 997

shortcuts=**abother**
Implements shortcuts=abbreviations and shortcuts=other.

20. *997*

shortcuts=**ac**
Implements the acronym shortcuts that are compatible with \newabbreviation.

Table 4.1; 19, 20, 44, 57, 700–702, 705, 849, 968, *997*

`shortcuts=`**acother** — 20, 849, *998*
Implements `shortcuts=ac` and `shortcuts=other`.

`shortcuts=`**acro** *alias:* `acronyms` — *see* `acronyms`

`shortcuts=`**acronyms** — 20, 700–702, 705, *998*
Implements the acronym shortcuts. Don't use this option unless you have reverted `\newacronym` back to the base glossaries package's acronym mechanism.

`shortcuts=`**all** — 20, 21, 849, 850, *998*
Implements `shortcuts=ac`, `shortcuts=abbreviations`, `shortcuts=other`.

`shortcuts=`**false** *alias:* `all` — *see* `all`

`shortcuts=`**none** — 21, *998*
Don't define any shortcut commands.

`shortcuts=`**other** — 19, 850, 970, 971, *998*
Implements the shortcuts `\newentry`, `\newsym` and `\newnum`.

`shortcuts=`**true** *alias:* `all` — *see* `all`

showtargets=⟨*value*⟩ glossaries−extra v1.48+ — §2.5; 30, 389, 925, *998*
Implements `debug=showtargets`.

`showtargets=`**annoteleft** — 31, 348, *998*
Markers are placed on either side of the link/target with the annotation on the left in all modes.

`showtargets=`**annoteright** — 32, *998*
Markers are placed on either side of the link/target with the annotation on the right in all modes.

`showtargets=`**innerleft** — 31, *998*
A marker and annotation are placed to the left of the link/target in all modes.

`showtargets=`**innerright** — 31, *998*
A marker and annotation are placed to the left of the link/target in all modes.

`showtargets=`**left** — 31, *998*
A marker is placed to the left of the link/target and a marginal note is used in outer mode.

`showtargets=`**right** — 31, *998*
A marker is placed to the right of the link/target and a marginal note is used in outer mode.

stylemods=⟨*value*⟩ *default:* default

Loads glossaries−extra−stylemods with the given options. If stylemods=default then no options are passed to glossaries−extra−stylemods.

stylemods=**all**
Load glossaries−extra−stylemods with all predefined styles.

stylemods=**default**
Load glossaries−extra−stylemods with no options (patches any predefined styles that have been loaded).

stylemods=⟨*list*⟩
Load glossaries−extra−stylemods with all the listed styles.

undefaction=⟨*value*⟩ *initial:* error

Indicates whether to trigger an error or warning if an unknown entry label is referenced.

undefaction=**error**
Trigger an error if an unknown entry label is referenced.

undefaction=**warn**
Trigger a warning if an unknown entry label is referenced.

\usepackage[⟨*options*⟩]{**glossaries**} (automatically loaded with \usepackage{glossaries-extra})

Base package. When loaded implicitly with glossaries−extra, any relevant options will be passed to the base glossaries package.

acronym=⟨*boolean*⟩ *default:* true; *initial:* false

If true, provides a new glossary with the label acronym and title given by \acronymname, redefines \acronymtype to acronym, and provides \printacronyms.

acronymlists={⟨*label-list*⟩}

Identifies the glossaries that contain acronyms (defined with the base glossaries packages acronym mechanism).

acronyms

Provides a new glossary with the label acronym, redefines \acronymtype to acronym, and provides \printacronyms.

automake=⟨*value*⟩ *default:* true; *initial:* false

Indicates whether or not to attempt to use TEX's shell escape to run an indexing application.

counter=⟨*counter-name*⟩ *initial:* page

Sets the default location counter.

debug=⟨*value*⟩ *initial:* `false` glossaries v4.24+ *1000*
(modified by glossaries−extra's `debug` option)
Adds markers to the document for debugging purposes.

entrycounter 385, 429, 431, 462, 471, 753, 806, *1000*
Enables the entry counter for top-level entries.

hyperfirst=⟨*boolean*⟩ *default:* `true`; *initial:* `true` 190, 261, 530, *1000*
Indicates whether or not to use hyperlinks on first use (if hyperlinks are supported).

index §2.1; 12, 13, 439, 581, 906, 971, 979, 982, *1000*
Provides a new glossary with the label `index` and the title `\indexname`, and provides `\printindex` and `\newterm`.

indexonlyfirst=⟨*boolean*⟩ *default:* `true`; *initial:* `false` 273, 530, 626, 949, *1000*
Indicates whether or not to only index the first use.

makeindex 605, *1000*, 1003
Indicates that the indexing should be performed by `makeindex` (default).

mfirstuc glossaries v4.50+ *1000*
Implements the `expanded` and `unexpanded` options provided by mfirstuc.

nogroupskip=⟨*boolean*⟩ *default:* `true`; *initial:* `false` 445–447, 484, 773, 802, 949, *1000*
If true, suppress the gap between letter groups in the glossaries by default.

nolist 15, *1000*
Don't load glossary−list, which is normally loaded automatically. Note that if glossaries is loaded before glossaries−extra, then this option should be passed directly to glossaries not glossaries−extra otherwise it will be too late to implement.

nolong *1000*
Don't load glossary−long, which is normally loaded automatically. Note that if glossaries is loaded before glossaries−extra, then this option should be passed directly to glossaries not glossaries−extra otherwise it will be too late to implement.

nomain 7, 748, *1000*
Prevents the definition of the `main` glossary. You will need to define another glossary to use instead. For example, with the `acronyms` package option.

nonumberlist 269, 410, 439, 440, 484, 575, 651, 723, 801, *1000*
Set no location lists as the default for all glossaries. May be overridden for individual glossaries with `nonumberlist=true`.

§2.2; 2–4, 13, 14, 445, 804, 996, *1001*, *see also* postdot & postpunc

nopostdot=⟨*boolean*⟩ *default:* `true`; *initial:* `true`
If true, suppresses the automatic insertion of a full stop after each entry's description in the glossary (for styles that support this). The default is `nopostdot=true` for glossaries−extra and `nopostdot=false` for just glossaries.

3, *1001*

noredefwarn
Suppresses a warning if theglossary or `\printglossary` have already been defined (which indicates that the document class or another package also provides a mechanism for creating a glossary that could potentially conflict with glossaries). This option is automatically implemented with glossaries−extra.

15, 444, 449, 453, *1001*

nostyles
Don't load the default set of predefined styles. Note that if glossaries is loaded before glossaries−extra, then this option should be passed directly to glossaries not glossaries−extra otherwise it will be too late to implement.

1001

nosuper
Don't load glossary−super, which is normally loaded automatically. Note that if glossaries is loaded before glossaries−extra, then this option should be passed directly to glossaries not glossaries−extra otherwise it will be too late to implement.

1001

notranslate
Equivalent to `translate=false`.

449, *1001*

notree
Don't load glossary−tree, which is normally loaded automatically. Note that if glossaries is loaded before glossaries−extra, then this option should be passed directly to glossaries not glossaries−extra otherwise it will be too late to implement.

721, *1001*

nowarn
Suppresses warnings.

387, 388, 498, 673, 723, §2.1; 12, 20, 33, 439, 581, 801, 900, 906, 971, 980, 983, *1001*

numberedsection=⟨*value*⟩ *default:* `nolabel`; *initial:* `false`
Indicates whether or not glossary section headers will be numbered and also if they should automatically be labelled.

numbers
Provides a new glossary with the label `numbers` and the title `\glsnumbersgroupname`, and provides `\printnumbers`. With glossaries−extra, this additionally defines `\glsxtrnewnumber`.

383, *1001*

sanitizesort=⟨*boolean*⟩ *default:* `true`; *initial: varies*
Indicates whether the default sort value should be sanitized (only applicable with `sort=standard`).

405, 434, 807, 832, *1001*

saveglossarygroups=⟨*boolean*⟩ *default:* `true`; *initial:* `false` glossaries v5.1+

If true, redefines `\glswriteglossarygroup` and `\glswriteglossarysubgroup` so that they write group information to the `aux` file. (The glossary style will need to support this). If false, those commands are redefined to do nothing.

savenumberlist=⟨*boolean*⟩ *default:* `true`; *initial:* `false` 556, *1002*
If true, patches the location list encapsulator to save the location list. With `bib2gls`, use the `save-locations` resource option.

section=⟨*value*⟩ *default:* `section` 459, 723, *1002*
Indicates which section heading command to use for the glossary. The value may be one of the standard sectioning command's control sequence name (without the leading backslash), such as `chapter` or `section`.

seenoindex=⟨*value*⟩ *initial:* `error` 24, *1002*
Indicates what to do if the `see` key is used before the associated indexing files have been opened by `\makeglossaries`.

`seenoindex`=**error** *1002*
Triggers an error if the `see` key is used before `\makeglossaries`.

`seenoindex`=**ignore** *1002*
Does nothing if the `see` key is used before `\makeglossaries`.

`seenoindex`=**warn** *1002*
Triggers a warning if the `see` key is used before `\makeglossaries`.

sort=⟨*value*⟩ *initial:* `standard` 399, 428, *1002*
Indicates how the `sort` key should automatically be assigned if not explicitly provided (for `\makeglossaries` and `\makenoidxglossaries` only).

`sort`=**clear** glossaries v4.50+ *1002*
Sets the `sort` key to an empty value. Use this option if no indexing is required for a slightly faster build.

`sort`=**def** *1002*
Use the (zero-padded) order of definition as the default for the `sort` key.

`sort`=**none** glossaries v4.30+ 25, *1002*
Don't process the `sort` key. Use this option if no indexing is required for a slightly faster build.

`sort`=**standard** 805, 1001, *1002*
Use the value of the `name` key as the default for the `sort` key and implement the `\glsprestandardsort` hook.

`sort`=**use** *1002*

Use the (zero-padded) order of use as the default for the `sort` key.

style=⟨*style-name*⟩ 491, *1003*
Sets the default glossary style to ⟨*style-name*⟩.

subentrycounter 385, 429, 431. 462. §2.1; 11, 19, 33, 439, 581, 610, 813, 901, 906, 971, 980, 983, *1003*
Enables the entry counter for level 1 entries.

symbols
Provides a new glossary with the label `symbols` and the title `\glssymbolsgroupname`, and provides `\printsymbols`. With glossaries−extra, this additionally defines `\glsxtrnewsymbol`.

toc=⟨*boolean*⟩ *default:* `true`; *initial:* `true` 2–4, *1003*
If true, each glossary will be automatically added to the table of contents. The default is `toc=false` with glossaries and `toc=true` with glossaries−extra.

translate=⟨*value*⟩ *initial:* `babel` 3, 4, 1001, *1003*
Indicates how multilingual support should be provided, if applicable.

`translate=`**babel** *1003*
Uses babel's language hooks to implement multilingual support (default for glossaries−extra if babel has been detected).

`translate=`**false** *1003*
Don't implement multilingual support (default if no language package has been detected).

`translate=`**true** *1003*
Uses translator's language hooks to implement multilingual support (default for glossaries if a language package has been detected).

upmendex glossaries v5.1+ 383, 541, *1003*
Indicates that the indexing should be performed by `upmendex`. This mostly behaves like the `makeindex` option.

writeglslabelnames 18, *1003*
Creates a file called `\jobname.glslabels` that contains all defined entry labels and names (for the benefit of autocompletion tools).

writeglslabels 18, *1003*
Creates a file called `\jobname.glslabels` that contains all defined entry labels (for the benefit of autocompletion tools).

xindy={⟨*options*⟩} 269, 271, 442, 624, *1003*
Indicates that the indexing should be performed by `xindy`.

Index

Symbols

A

B

C

D

E

F

G

Glo

Gls

Glsxtr

H

I

M

N

O

P

Q

R

S

T

U

W

X

Z