Upgrading from the glossary package to the glossaries package

Nicola L.C. Talbot

2025-05-14

This document is also available as HTML (`glossary2glossaries.html`).

> The purpose of this document is to provide advice if you want to convert a LaTeX document from using the obsolete glossary package to the replacement glossaries package. The final version of the glossary package is 2.4 (2006-07-20). It was made obsolete after the release of glossaries v1.0 (2007-05-16).

For the main glossaries user guide, see `glossaries-user.pdf`.

```
texdoc glossaries-user
```

For a shorter guide for beginners, see `glossariesbegin.pdf`.

```
texdoc glossariesbegin
```

Contents

1 Why the Need for a New Package?

The glossary package started out as an example in a tutorial, but I decided that I may as well package it up and upload it to CTAN. Unfortunately it was fairly rigid and unable to adapt well to the wide variation in glossary styles. Users began making requests for enhancements, but with each enhancement the code became more complicated and bugs crept in. Each fix in one place seemed to cause another problem elsewhere. In the end, it was taking up too much of my time to maintain, so I decided to replace it with a much better designed package. With the new glossaries package:

- you can define irregular plurals;
- glossary terms can have an associated symbol in addition to the name and description;
- new glossary styles are much easier to design;
- you can add dictionaries to supply translations for the fixed names used in headings and by some of the glossary styles;
- you can choose between using `makeindex` or `xindy` to sort the glossary. Using `xindy` means that:
  - there is much better support for terms containing accented or non-Latin characters;
  - there is support for non-standard location numbers;

(Additional indexing options have since been added. See the "Indexing Options" section of the glossaries user manual.)

- you don't need to remember to escape `makeindex`'s special characters as this is done internally;
- hierarchical entries and homographs are supported (as from v1.17);
- there is better support for cross-referencing glossary entries;
- acronyms are just another glossary term which helps to maintain consistency;
- different acronym styles are supported.

2 Package Options

When converting a document that currently uses the obsolete glossary package to the replacement glossaries package, it should be fairly obvious that the first thing you need to do is replace `\usepackage{glossary}` with `\usepackage{glossaries}`, however some of the package options are different, so you may need to change those as well. Table 1 shows the mappings from the glossary to the glossaries package options.

3 Defining new glossary types

If you have created new glossary types, you will need to replace all instances of

glossary.sty

```
\newglossarytype[⟨log-ext⟩]{⟨type⟩}{⟨out-ext⟩}{⟨in-ext⟩}[⟨old style list⟩]

\newcommand{⟨type⟩name}{⟨title⟩}
```

with

glossaries.sty

```
\newglossary[⟨log-ext⟩]{⟨type⟩}{⟨out-ext⟩}{⟨in-ext⟩}{⟨title⟩}
```

in the preamble, and, if the new glossary requires a different style to the main (default) glossary, you will also need to put

glossaries.sty

```
\setglossarystyle{⟨new-style⟩}
```

immediately before the glossary is displayed, or you can specify the style when you display the glossary using `\printglossary` (see below).

Table 1: Mappings from glossary to glossaries package options

glossary option	**glossaries option**
style=list	style=list
style=altlist	style=altlist
style=long,header=none,border=none,cols=2	`style`=long
style=long,header=plain,border=none,cols=2	`style`=longheader
style=long,header=none,border=plain,cols=2	`style`=longborder
style=long,header=plain,border=plain,cols=2	`style`=longheaderborder
style=long,header=none,border=none,cols=3	`style`=long3col
style=long,header=plain,border=none,cols=3	`style`=long3colheader
style=long,header=none,border=plain,cols=3	`style`=long3colborder
style=long,header=plain,border=plain,cols=3	`style`=long3colheaderborder
style=super,header=none,border=none,cols=2	`style`=super
style=super,header=plain,border=none,cols=2	`style`=superheader
style=super,header=none,border=plain,cols=2	`style`=superborder
style=super,header=plain,border=plain,cols=2	`style`=superheaderborder
style=super,header=none,border=none,cols=3	`style`=super3col
style=super,header=plain,border=none,cols=3	`style`=super3colheader
style=super,header=none,border=plain,cols=3	`style`=super3colborder
style=super,header=plain,border=plain,cols=3	`style`=super3colheaderborder
number=none	`nonumberlist`
number=⟨*counter name*⟩	`counter`=⟨*counter name*⟩
toc	`toc`
hypertoc	`toc`
hyper	*no corresponding option*
section=true	`section`
section=false	*no corresponding option*
acronym	`acronym`
global	*no corresponding option*

The ⟨*old style list*⟩ optional argument can be converted to ⟨*new-style*⟩ using the same mapping given in Table 1.

For example, if your document contains the following:

glossary.sty
```
\newglossarytype[nlg]{notation}{not}{ntn}
[style=long,header]
\newcommand{\notationname}{Index of Notation}
```

You will need to replace the above two lines with:

glossaries.sty
```
\newglossary[nlg]{notation}{not}{ntn}
{Index of Notation}
```

in the preamble and set the style to longheader with

glossaries.sty
```
\setglossarystyle{longheader}
```

prior to displaying this glossary. Alternatively, you can specify the style using `style` package option (which makes it the default style) or the `style` key in the optional argument of `\printglossary`. For example:

glossaries.sty
```
\printglossary[type=notation,style=longheader]
```

Note that the glossary title is no longer specified using `\`⟨*glossary−type*⟩`name` (except for `\glossaryname` and `\acronymname`) but is instead specified in the ⟨*title*⟩ argument of `\newglossary`. The short title which is specified in the glossary package by the command `\short`⟨*glossary−type*⟩`name` is now specified using the `toctitle` key in the optional argument to `\printglossary`.

4 `\make`⟨*glossary−type*⟩

All instances of `\make`⟨*glossary−type*⟩ (e.g. `\makeglossary` and `\makeacronym`) should be replaced by the single command `\makeglossaries`. For example, if your document contained the following:

glossary.sty
```
\makeglossary
\makeacronym
```

then you should replace both lines with the single line:

glossaries.sty

```
\makeglossaries
```

5 Storing glossary information

With the old glossary package you could optionally store glossary information for later use, or you could simply use `\glossary` whenever you wanted to add information to the glossary. With the new glossaries package, the latter option is no longer available. (This is mainly because having a key value list in `\glossary` caused problems, but it also helps consistency.) If you have stored all the glossary information using `\storeglosentry`, then you will need to convert these commands into the equivalent `\newglossaryentry`. If you have only used `\glossary`, then see §6.4.

Substitute all instances of

glossary.sty

```
\storeglosentry{⟨label⟩}{⟨key=value list⟩}
```

with

glossaries.sty

```
\newglossaryentry{⟨label⟩}{⟨key=value list⟩}
```

This should be fairly easy to do using the search and replace facility in your editor (but see notes below).

If you have used the optional argument of `\storeglosentry` (i.e. you have multiple glossaries) then you will need to substitute

glossary.sty

```
\storeglosentry[⟨gls-type⟩]{⟨label⟩}{⟨key=value list⟩}
```

with

glossaries.sty

```
\newglossaryentry{⟨label⟩}{⟨key=value list⟩,type={⟨gls-type⟩}}
```

The glossary entry information ⟨*key=value list*⟩ may also need changing. If ⟨*key=value list*⟩ contains any of `makeindex`'s special characters (i.e. `@ ! "` or `|`) then they should no longer be escaped with `"` since the glossaries package deals with these characters internally. For example, if your document contains the following:

glossary.sty

```
\storeglosentry{card}{name={$"|\mathcal{S}"|$},
description={The cardinality of the set $\mathcal{S}
$}}
```

then you will need to replace it with:

glossaries.sty

```
\newglossaryentry{card}{name={$|\mathcal{S}|$},
description={The cardinality of the set $\mathcal{S}
$}}
```

The `number` key available in `\storeglosentry` should be replaced with the `counter` key in `\newglossaryentry`. The `sort` key in `\storeglosentry` is also called `sort` in `\newglossaryentry`.

The `\storeglosentry` `format` key doesn't have a counterpart in `\newglossaryentry`. You can, however, specify the format in the optional argument of commands like `\gls` or `\glsadd` or you can change the default format by redefining `\glsnumberformat`.

6 Adding an entry to the glossary

The glossary package provided two basic means to add information to the glossary: firstly, the term was defined using `\storeglosentry` and the entries for that term were added using `\useglosentry`, `\useGlosentry` and `\gls`. Secondly, the term was added to the glossary using `\glossary`. This second approach is unavailable with the glossaries package, since all entries must be defined before they can be indexed.

6.1 `\useglosentry`

The glossary package allows you to add information to the glossary for a predefined term without producing any text in the document using

glossary.sty

```
\useglosentry[⟨old options⟩]{⟨label⟩}
```

Any occurrences of this command will need to be replaced with

glossaries.sty

```
\glsadd[⟨new options⟩]{⟨label⟩}
```

The `format` key in ⟨*old options*⟩ is also called `format` in ⟨*new options*⟩. However the `number` `=`{⟨*counter-name*⟩} key in ⟨*old options*⟩ should be replaced with `counter` `=`

⟨*counter-name*⟩ in ⟨*new options*⟩.

6.2 \useGlosentry

The glossary package allows you to add information to the glossary for a predefined term with the given text using

glossary.sty

```
\useGlosentry[⟨old options⟩]{⟨label⟩}{⟨text⟩}
```

Any occurrences of this command will need to be replaced with

glossaries.sty

```
\glslink[⟨new options⟩]{⟨label⟩}{⟨text⟩}
```

The mapping from ⟨*old options*⟩ to ⟨*new options*⟩ is the same as that given §6.1.

6.3 \gls

The glossary defines:

glossary.sty

```
\gls (glossary.sty)[⟨options⟩]{⟨label⟩}
```

The glossaries package defines a command with the same name, but be aware that it has a final optional argument:

glossaries.sty

```
\gls (glossaries.sty)[⟨options⟩]{⟨label⟩}[⟨insert⟩]
```

In this case, the only thing you need to change is the `number` key in the optional argument to `counter`. The ⟨*insert*⟩ optional argument in the new form of `\gls` can be used to insert text into the automatically generated text, which will put it inside the hyperlink (if hyperlinks are supported).

6.4 \glossary

When using the glossaries package, you should not use `\glossary`. This is because the appropriate indexing syntax (including escaping any of `makeindex`'s or `xindy`'s special characters) is generated when the entry is defined. This reduces overall complexity as it no longer needs to be performed every time an entry is indexed. By placing the glossary definitions within the preamble, it also reduces the chance that the indexing special character may have their category code changed, which can cause interference.

If, with the old package, you have opted to explicitly use `\glossary` instead of storing the glossary information with `\storeglosentry`, then converting from glossary to glossaries will be more time-consuming, although in the end, I hope you will see the benefits. From the user's point of view, using `\glossary` throughout the document is time consuming, and if you use it more than once for the same term, there's a chance extra spaces may creep in which will cause `makeindex` to treat the two entries as different terms, even though they look the same in the document. If you have used `\glossary` with the old glossary package, you will instead need to define the relevant glossary terms using `\newglossaryentry` and reference the terms using `\glsadd`, `\glslink`, `\gls` etc.

If you don't like the idea of continually scrolling back to the preamble to type all your `\newglossaryentry` commands, you may prefer to create a new file, in which to store all these commands, and then input that file in your document's preamble. Most text editors and front-ends allow you to have multiple files open, and you can tab back and forth between them.

7 Acronyms

In the glossary package, acronyms were treated differently to glossary entries. This resulted in inconsistencies and sprawling unmaintainable code. The new glossaries package treats acronyms in exactly the same way as normal glossary terms.

Both packages provide `\newacronym`, but the syntax is different. With the glossary package, the syntax is:

glossary.sty

```
\newacronym
(glossary.sty)[⟨cmd-name⟩]{⟨acronym⟩}{⟨long⟩}{⟨old-options⟩}
```

With the glossaries package, the default definition of:

glossaries.sty

```
\newacronym
(glossaries.sty)[⟨options⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}
```

is a shortcut for:

glossaries.sty

```
\newglossaryentry{⟨label⟩}{type=\acronymtype,
name={⟨abbrv⟩},
description={⟨long⟩},
text={⟨abbrv⟩},
first={⟨long⟩ (⟨abbrv⟩)},
plural={⟨abbrv⟩s},
firstplural={⟨long⟩s (⟨abbrv⟩s)},
```

```
⟨options⟩}
```

(Note that this shortcut default is an older method of defining acronyms. If you use `\setacronymstyle` introduced to glossaries v4.02, then a more flexible method is adopted.)

This is different to the glossary package which set the `name` key to ⟨*long*⟩ (⟨*abbrv*⟩) and allowed you to set a description using the `description` key. If you still want to do this, you can use one of the description styles, such as long−short−desc, and use the `description` key in the optional argument of `\newacronym`.

For example, if your document originally had the following:

glossary.sty

```
\newacronym{SVM}{Support Vector Machine}{description
={Statistical
pattern recognition technique}}
```

Then you would need to first set the style:

glossaries.sty

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

and change the acronym definition to:

glossaries.sty

```
\newacronym[description=
{Statistical pattern recognition
technique}]{svm}{SVM}{Support Vector Machine}
```

You can then reference the acronym using any of the new referencing commands, such as `\gls` or `\glsadd`.

With the old glossary package, when you defined an acronym, it also defined a command `\`⟨*acr−name*⟩ which could be used to display the acronym in the text. So the above SVM example would create the command `\SVM` with the old package. In the new glossaries package, the acronyms are just another type of glossary entry, so they are displayed using `\gls{`⟨*label*⟩`}`. Therefore, in the above example, you will also need to replace all occurrences of `\SVM` with `\gls{svm}`.

If you have used `\useacronym` instead of `\`⟨*acr−name*⟩, then you will need to replace all occurrences of

glossary.sty

```
\useacronym[⟨insert⟩]{⟨acr-name⟩}
```

with

glossaries.sty

```
\gls{⟨label⟩}[⟨insert⟩]
```

Note that the starred versions of \useacronym and \⟨*acr−name*⟩ (which make the first letter uppercase) should be replaced with \Gls{⟨*label*⟩}.

Alternatively (as from v1.18 of the glossaries package), you can use \oldacronym which uses the same syntax as the old glossary package's \newacronym and also defines \⟨*acr−name*⟩. For example, if your document originally had the following:

glossary.sty

```
\newacronym{SVM}{Support Vector Machine}{description
={Statistical
pattern recognition technique}}
```

then you can change this to:

glossaries.sty

```
\oldacronym{SVM}{Support Vector Machine}{description
={Statistical
pattern recognition technique}}
```

You can then continue to use \SVM. However, remember that LaTeX generally ignores spaces after command names that consist of alphabetical characters. You will therefore need to force a space after \⟨*acr−name*⟩, unless you also load the xspace package. (See the "Acronyms" of the glossaries documentation for further details.) Note that \oldacronym uses its first argument to define the acronym's label (as used by commands like \gls), so in the above example, with the new glossaries package, \SVM becomes a shortcut for \gls{SVM} and \SVM* becomes a shortcut for \Gls{SVM}.

7.1 \acrln and \acrsh

In the glossary package, it is possible to produce the long and short forms of an acronym without adding an entry to the glossary using \acrln and \acrsh. With the glossaries package (provided you defined the acronym using \newacronym or \oldacronym and provided you haven't redefined \newacronym) you can replace

glossary.sty

```
\acrsh{⟨acr-name⟩}
```

with

glossaries.sty

```
\acrshort{⟨label⟩}
```

and you can replace

glossary.sty

```
\acrln{⟨acr-name⟩}
```

with

glossaries.sty

```
\acrlong{⟨label⟩}
```

The glossaries package also provides the related commands `\acrshortpl` (plural short form) and `\acrlongpl` (plural long form) as well as upper case variations. If you use the glossaries "shortcuts" package option, you can use `\acs` in place of `\acrshort` and `\acl` in place of `\acrlong`.

See the "Acronyms" of the glossaries manual for further details of how to use these commands.

7.2 `\ifacronymfirstuse`

The glossary package command

glossary.sty

```
\ifacronymfirstuse{⟨acr-name⟩}{⟨not used text⟩}{⟨has been used text⟩}
```

can be replaced by the glossaries command:

glossaries.sty

```
\ifglsused{⟨label⟩}{⟨has been used text⟩}{⟨not used text⟩}
```

Note that `\ifglsused` evaluates the opposite condition to that of `\ifacronymfirstuse` which is why the last two arguments have been reversed.

7.3 `\resetacronym` and `\unsetacronym`

The glossary package allows you to reset and unset the acronym flag which is used to determine whether the acronym has been used in the document. The glossaries package also provides a means to do this on either a local or a global level. To reset an acronym, you will need to replace:

glossary.sty

```
\resetacronym{⟨acr-name⟩}
```

with either

glossaries.sty

```
\glsreset{⟨label⟩}
```

or

glossaries.sty

```
\glslocalreset{⟨label⟩}
```

To unset an acronym, you will need to replace:

glossary.sty

```
\unsetacronym{⟨acr-name⟩}
```

with either

glossaries.sty

```
\glsunset{⟨label⟩}
```

or

glossaries.sty

```
\glslocalunset{⟨label⟩}
```

To reset all acronyms, you will need to replace:

glossary.sty

```
\resetallacronyms
```

with

glossaries.sty

```
\glsresetall[\acronymtype]
```

or

glossaries.sty

```
\glslocalresetall[\acronymtype]
```

To unset all acronyms, you will need to replace:

glossary.sty

```
\unsetallacronyms
```

with

glossaries.sty
```
\glsunsetall[\acronymtype]
```
or
glossaries.sty
```
\glslocalunsetall[\acronymtype]
```

8 Displaying the glossary

The glossary package provides the command `\printglossary` (or `\print`⟨*type*⟩ for other glossary types) which can be used to print individual glossaries. The glossaries package provides the command `\printglossaries` which will print all the glossaries which have been defined, or `\printglossary` (glossaries.sty) `[`⟨*options*⟩`]` to print individual glossaries. So if you just have `\printglossary`, then you can leave it as it is, but if you have, say:

glossary.sty
```
\printglossary
\printglossary[acronym]
```
or
glossary.sty
```
\printglossary
\printacronym*
```
then you will need to replace this with either
glossaries.sty
```
\printglossaries
```
or
glossaries.sty
```
\printglossary
\printglossary[type=\acronymtype]
```

The glossary package allows you to specify a short title (for the table of contents and page header) by defining a command of the form `\short`⟨*glossary−type*⟩`name`. The glossaries package doesn't do this, but instead provides the `toctitle` key which can be used in the optional argument to `\printglossary`. For example, if you have created a new glossary type called `notation`, and you had defined

glossary.sty

```
\newcommand{\shortnotationname}{Notation}
```

then you would need to use the `toctitle` key:

glossaries.sty

```
\printglossary[type=notation,toctitle=Notation]
```

The glossaries package will ignore `\shortnotationname`, so unless you have used it elsewhere in the document, you may as well remove the definition.

9 Processing Your Document

If you convert your document from using the glossary package to the glossaries package, you will need to delete any of the additional files, such as the `glo` file, that were created by the glossary package, as the glossaries package uses a different format. Remember also, that if you used the `makeglos` Perl script, you will need to use the `makeglossaries` Perl script instead. As from v1.17, the glossaries package can be used with either `makeindex` or `xindy`. Since `xindy` was designed to be multilingual, the new glossaries package is a much better option for non-English documents. If you use the extension package, glossaries-extra, then you also have the option of using `bib2gls` instead (which also provides multilingual support).

For further information on using `makeglossaries`, `makeindex` or `xindy` to create your glossaries, see the "Generating the Associated Glossary Files" section of the glossaries documentation.

10 Troubleshooting

Please check the FAQ[1] for the glossaries package if you have any problems.

Symbols

Symbol	Description
>_	A command-line application invocation that needs to be entered into a terminal or command prompt.

Index

[1] `dickimaw-books.com/faqs/glossariesfaq.html`

Symbols

A

B

C

F

G

I

M

N

O

P

R

S

T

U

X