

Upgrading from the glossary package to the glossaries package

Nicola L.C. Talbot

14th January 2009

Contents

1	Why the Need for a New Package?	2
2	Package Options	2
3	Defining new glossary types	4
4	<code>\make</code><i><glossary name></i>	4
5	Storing glossary information	5
6	Adding an entry to the glossary	6
6.1	<code>\usegloentry</code>	6
6.2	<code>\useGloentry</code>	6
6.3	<code>\gls</code>	7
6.4	<code>\glossary</code>	7
7	Acronyms	7
7.1	<code>\acrln</code> and <code>\acrsh</code>	9
7.2	<code>\ifacronymfirstuse</code>	10
7.3	<code>\resetacronym</code> and <code>\unsetacronym</code>	10
8	Displaying the glossary	11
9	Processing Your Document	12
10	Troubleshooting	12

Abstract

The purpose of this document is to provide advice if you want to convert a L^AT_EX document from using the obsolete `glossary` package to the replacement `glossaries` package.

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 choose¹ 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;
- you don't need to remember to escape `makeindex`'s special characters as this is done internally;
- hierarchical entries and homographs are supported;²
- 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.

¹as from v1.17. Previous versions were designed to be used with `makeindex` only

²as from v1.17

Table 1: Mappings from glossary to glossaries package options

glossary option

style=list
style=altlist
style=long,header=none,border=none,cols=2
style=long,header=plain,border=none,cols=2
style=long,header=none,border=plain,cols=2
style=long,header=plain,border=plain,cols=2
style=long,header=none,border=none,cols=3
style=long,header=plain,border=none,cols=3
style=long,header=none,border=plain,cols=3
style=long,header=plain,border=plain,cols=3
style=super,header=none,border=none,cols=2
style=super,header=plain,border=none,cols=2
style=super,header=none,border=plain,cols=2
style=super,header=plain,border=plain,cols=2
style=super,header=none,border=none,cols=3
style=super,header=plain,border=none,cols=3
style=super,header=none,border=plain,cols=3
style=super,header=plain,border=plain,cols=3
number=none
number=*\langle counter name \rangle*
toc
hypertoc
hyper
section=true
section=false
acronym
global

glossaries option

style=list
style=altlist
style=long
style=longheader
style=longborder
style=longheaderborder
style=long3col
style=long3colheader
style=long3colborder
style=long3colheaderborder
style=super
style=superheader
style=superborder
style=superheaderborder
style=super3col
style=super3colheader
style=super3colborder
style=super3colheaderborder
nonumberlist
counter=*\langle counter name \rangle*
toc
toc
no corresponding option
section
no corresponding option
acronym
no corresponding option

3 Defining new glossary types

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

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

glossary

with

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

glossaries

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

```
\glossarystyle{⟨new style⟩}
```

glossaries

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

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:

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

You will need to replace the above two lines with:

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

in the preamble and

```
\glossarystyle{longheader}
```

immediately prior to displaying this glossary. Alternatively, you can specify the style using the `style` key in the optional argument of `\printglossary`. For example:

```
\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-name⟩name` is now specified using the `toctitle` key in the optional argument to `\printglossary`.

4 `\make⟨glossary name⟩`

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

```
\makeglossary
\makeacronym
```

then you should replace both lines with the single line:

```
\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.³ If you have stored all the glossary information using `\storegloentry`, then you will need to convert these commands into the equivalent `\newglossaryentry`. If you have only used `\glossary`, then see section 6.4.

Substitute all instances of

```
\storegloentry{<label>}{<gls-entry>}
```

`glossary`

with

```
\newglossaryentry{<label>}{<gls-entry>}
```

`glossaries`

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 `\storegloentry` (i.e. you have multiple glossaries) then you will need to substitute

```
\storegloentry[<gls-type>]{<label>}{<gls-entry>}
```

`glossary`

with

```
\newglossaryentry{<label>}{<gls-entry>,type=<gls-type>}
```

`glossaries`

The glossary entry information `<gls-entry>` may also need changing. If `<gls-entry>` 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:

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

³mainly because having a key value list in `\glossary` caused problems, but it also helps consistency.

then you will need to replace it with:

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

The format and number keys available in `\storeglosentry` are not available with `\newglossaryentry`.

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.

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

```
\useglosentry[<old options>]{<label>}
```

`glossary`

Any occurrences of this command will need to be replaced with

```
\glsadd[<new options>]{<label>}
```

`glossaries`

The `format` key in *<old options>* remains the same 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

```
\useGlosentry[<old options>]{<label>}{<text>}
```

`glossary`

Any occurrences of this command will need to be replaced with

```
\glslink[<new options>]{<label>}{<text>}
```

`glossaries`

The mapping from *<old options>* to *<new options>* is the same as that given in section 6.1 above.

6.3 `\gls`

Both the `glossary` and the `glossaries` packages define the command `\gls`. In this case, the only thing you need to change is the `number` key in the optional argument to `counter`. Note that the new form of `\gls` also takes a final optional argument which can be used to insert text into the automatically generated text.

6.4 `\glossary`

When using the `glossaries` package, you should not use `\glossary` directly.⁴ If, with the old package, you have opted to explicitly use `\glossary` instead of storing the glossary information with `\storegloentry`, then converting from `glossary` to `glossaries` will be more time-consuming, although in the end, I hope you will see the benefits.⁵ 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. In fact, in the `glossaries` package, the default definition of:

`\newacronym[<options>]{<label>}{<abbrv>}{<long>}`

`glossaries`

is a shortcut for:

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

`glossaries`

⁴This is because `\glossary` requires the argument to be in a specific format and doesn't use the `<key>=<value>` format that the old `glossary` package used. The new package's internal commands set this format, as well as escaping any of `makeindex`'s or `xindy`'s special characters. What's more, the format has changed as from v1.17 to allow the new package to be used with either `makeindex` or `xindy`.

⁵From 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.

This is different to the `glossary` package which set the `name` key to `\langle long \rangle` (`\langle abbrev \rangle`) and allowed you to set a description using the `description` key. If you still want to do this, you can use the `description` package option, and use the `description` key in the optional argument of `\newacronym`.

For example, if your document originally had the following:

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

Then you would need to load the `glossaries` package using the `description` package option, for example:

```
\usepackage[description]{glossaries}
```

and change the acronym definition to:

```
\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 `\langle acr-name \rangle` 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 `\langle acr-name \rangle`, then you will need to replace all occurrences of

`\useacronym[\langle insert \rangle]{\langle acr-name \rangle}`

glossary

with

`\gls{\label}[\langle insert \rangle]`

glossaries

Note that the starred versions of `\useacronym` and `\langle acr-name \rangle` (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 `\langle acr-name \rangle`. For example, if your document originally had the following:

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

glossary

then you can change this to:


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

glossaries

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 `\langle acr-name \rangle`, unless you also load the `xspace` package. (See section 2.10 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

```
\acrsh{\acr-name}
```

glossary

with

```
\acrshort{\label}
```

glossaries

and you can replace

```
\acrln{\acr-name}
```

glossary

with

```
\acrlong{\label}
```

glossaries

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 section 2.10 of the `glossaries` manual for further details of how to use these commands.

7.2 `\ifacronymfirstuse`

The `glossary` package command

```
\ifacronymfirstuse{<acr-name>}{<text1>}{<text2>}
```

`glossary`

can be replaced by the `glossaries` command:

```
\ifglused{<label>}{<text2>}{<text1>}
```

`glossaries`

Note that `\ifglused` 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:

```
\resetacronym{<acr-name>}
```

`glossary`

with either

```
\glsreset{<label>}
```

`glossaries`

or

```
\glslocalreset{<label>}
```

`glossaries`

To unset an acronym, you will need to replace:

```
\unsetacronym{<acr-name>}
```

`glossary`

with either

```
\glsunset{<label>}
```

`glossaries`

or

```
\glslocalunset{<label>}
```

glossaries

To reset all acronyms, you will need to replace:

```
\resetallacronyms
```

glossary

with

```
\glsresetall[<acronymtype>]
```

glossaries

or

```
\glslocalresetall[<acronymtype>]
```

glossaries

To unset all acronyms, you will need to replace:

```
\unsetallacronyms
```

glossary

with

```
\glsunsetall[<acronymtype>]
```

glossaries

or

```
\glslocalunsetall[<acronymtype>]
```

glossaries

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[<options>]` to print individual glossaries. So if you just have `\printglossary`, then you can leave it as it is, but if you have, say:

```
\printglossary  
\printglossary[acronym]
```

or

```
\printglossary
\printacronym
```

then you will need to replace this with either

```
\printglossaries
```

or

```
\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

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

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

```
\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.

For further information on using `makeglossaries`, `makeindex` or `xindy` to create your glossaries, see section 1.3 of the `glossaries` documentation.

10 Troubleshooting

Please check the FAQ⁶ for the `glossaries` package if you have any problems.

⁶<http://theoval.cmp.uea.ac.uk/nlct/latex/packages/faq/glossariesfaq.html>