

Upgrading from the glossary package to the glossaries package

Nicola L.C. Talbot

17th June 2008

Contents

1	Package Options	1
2	Defining new glossary types	1
3	<code>\make<glossary name></code>	3
4	Storing glossary information	3
5	Adding an entry to the glossary	4
5.1	<code>\usegloentry</code>	5
5.2	<code>\useGloentry</code>	5
5.3	<code>\gls</code>	5
5.4	<code>\glossary</code>	6
6	Acronyms	6
6.1	<code>\acrln</code> and <code>\acrsh</code>	8
6.2	<code>\ifacronymfirstuse</code>	8
6.3	<code>\resetacronym</code> and <code>\unsetacronym</code>	9
7	Displaying the glossary	9
8	Using <code>makeindex</code>	10
9	Troubleshooting	10

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 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.

Table 1: Mappings from `glossary` to `glossaries` package options

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

2 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>}[<style
list>]
\newcommand{\<type>name}{<title>}
```

`glossary`

with

<code>\newglossary[<log-ext>]{<type>}{<out-ext>}{<in-ext>}{<title>}</code>
--

glossaries

in the preamble, and

<code>\glossarystyle{<new style>}</code>
--

glossaries

immediately before `\printglossary[type=<type>]`, if the new glossary requires a different style to the main (default) glossary.

The `<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 new type.

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`.

3 `\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
```

4 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.¹ If you have stored all the glossary information using `\storeglosentry`, then you will need to convert these commands into the equivalent `\newglossaryentry`. If you haven't used `\storeglosentry`, then you'll have a bit more work to do!

The former approach requires substituting all instances of

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

`\storeglosentry[<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. `@ ! " |`) 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:

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

then you will need to replace it with:

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

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

¹mainly because having a key value list in `\glossary` caused problems, but it also helps consistency.

5 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 `\storegloentry` and the entries for that term were added using `\usegloentry`, `\useGloentry` and `\gls`. Secondly, the term was added to the glossary using `\glossary`. This second approach is unavailable with the `glossaries` package.

5.1 `\usegloentry`

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

`\usegloentry[<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>`, the `number=<counter name>` key in `<old options>` should be replaced with `counter=<counter name>` in `<new options>`.

5.2 `\useGloentry`

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

`\useGloentry[<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 section 5.1 above.

5.3 `\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.

5.4 `\glossary`

When using the `glossaries` package, you should not use `\glossary` directly.² If, with the old package, you have opted to explicitly use `\glossary` instead of storing the glossary information with `\storeglossentry`, then converting from `glossary` to `glossaries` will be more time-consuming, although in the end, I hope you will see the benefits!³ 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 `\glslink`, `\gls` (or `\glspl` 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.

6 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`

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 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
```

²This 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 special characters, so although it is still possible to use `\glossary` with the new package, it's not recommended. If you persist in using `\glossary` with the new package, don't complain if things go wrong!

³From 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.

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 will also need to replace all occurrences of `\SVM` with `\gls{svm}`. Alternatively, you can define `\SVM`:

```
\newcommand{\SVM}{\gls{svm}}
```

Remember that generally \LaTeX ignores all spaces following command names that consist of a backslash followed by letters! For example, using the above definition of `\SVM`, the following will ignore the space after `\SVM`:

The `\SVM` is a statistical pattern recognition technique.

Instead, you need to explicitly insert a space:

The `\SVM\` is a statistical pattern recognition technique.

However, `\SVM['s]` is equivalent to `\gls{svm}['s]`, so you can do, for example:

The `\SVM['s]` parameters are obtained via optimisation techniques.

The `xspace` package provides the command `\xspace` which can be used to insert a space after the command name if required. Some people prefer to use this at the end of the command definition so that they don't need to remember to insert an explicit space. For example, if we use the `xspace` package, we could define `\SVM` as follows:

```
\newcommand{\SVM}{\gls{svm}\xspace}
```

so now the following will have a space before the word “is”:

The `\SVM` is a statistical pattern recognition technique.

However, you can no longer do `\SVM['s]`, as the `\xspace` is now in the way.

If you have used `\useacronym` instead of `\<acr-name>`, then you will need to replace all occurrences of

`\useacronym[<insert>]{<acr-name>}`

glossary

with

`\gls{<label>}[<insert>]`

glossaries

Note that the starred versions of `\useacronym` and `\<acr-name>` (which make the first letter uppercase) should be replaced with `\Gls{<label>}`.

6.1 `\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` 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.

See section 5.10 (“Using glossary entries in the text”) of the `glossaries` manual for further details of how to use these commands.

6.2 `\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.

6.3 `\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 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`

7 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 either 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.

8 Using makeindex

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 `makeindex` style file. Remember also, that if you used the `makeglos` Perl script, you will need to use the `makeglossaries` Perl script instead.

9 Troubleshooting

Please check the FAQ⁴ for the `glossaries` package if you have any problems.

⁴<http://theoval.cmp.uea.ac.uk/nlct/latex/packages/faq/glossariesfaq.html>