

The `chemcompounds` package*

Stephan Schenk
mail@schenk-stephan.de

October 24, 2005

Abstract

The `chemcompounds.dtx` package allows for a simple consecutive numbering of chemical compounds. Optionally, it is possible to supply a custom name for each compound. By default the compound numbering follows the order of their appearance in the text.

Contents

1	Introduction	1
2	The user interface	2
3	The implementation	4

1 Introduction

In chemical publications it is often necessary to consecutively number every compound mentioned in the text. Although this can be simply accomplished by manually inserting the corresponding numbers into the text, it is generally much more tedious work since the numbering scheme tends to change several times during the evolution of the manuscript. For this reason it would be nice to have an automaticism which will take care of every change.

Being myself a chemist I've been using the `chemcono` package by Stefan Schulz for this purpose quite successfully over several years. This package creates a library very similar to `thebibliography`. Users can now refer to entries in the library by a command very similar to `\cite`. Thus, once you change the library entry every reference to it will be updated automatically upon running `LATEX` on the file. There is only one issue associated with this package: You get a list of all declared compounds inside your document which at least looks odd. I therefore decided to write a new package `chemcompounds` described in this document to address this problem.

When taking a closer look at the `chemcono` package I realised that the only thing one has to do is to get rid of everything which produces text. Thus, as a basis I used the mechanism of `\bibitem` and `\cite` in pretty much the same way as `chemcono` does by extracting the corresponding code from `article.cls`

*This file describes version v1.1.2 and has been last revised 2005/10/19.

and `latex.ltx` but deleting any unnecessary commands producing output. I also introduced several lines of code to make the printing of the compound names more customisable.

At present the packages knows two different modes of operation. In the default `implicit` mode a compound name is created automatically as a consecutive number when the compound is referenced for the first time by `\compound`. Thus, all compounds will be numbered consecutively in the order they appear in the text. If the automatically generated name is not appropriate, a custom name can be given to a compound by means of the `\declarecompound` command described below.

In `noimplicit` mode names are not generated fully automatically. Instead for every compound a `\declarecompound` must be issued. This will again create a subsequent number and a custom name can be given as an optional argument. The main difference to `implicit` mode is that thus compounds will be numbered in the order of the corresponding `\declarecompound` commands rather than in the order of their appearance in the text.

2 The user interface

Because of the way the implementation works, two \LaTeX runs are required to get everything right. This should not be a problem since you have to run \LaTeX twice anyway to get the table of contents and references right. The package will issue a “labels have changed” warning if you have to rerun \LaTeX . For every unknown compound name the package will issue a warning, too.

2.1 Package options

- `implicit` This option causes the package to operate in `implicit` mode. This is the default.
- `noimplicit` This option is the opposite of `implicit` and causes the package to operate in `noimplicit` mode.

2.2 Assigning and accessing compound names

- `\declarecompound` `\declarecompound[<optional name>]{<label>}`
assigns a name to a compound. If the optional argument is omitted, a consecutive number is automatically taken as compound name. This command can only occur in the preamble. A personal recommendation is to keep all `\declarecompounds` together in a separate file and `\input` this file in the preamble.

In `implicit` mode, if no optional argument is given, the command does nothing since the automatic compound name will be generated by `\compound`.

- `\compound` `\compound{<label1>,<label2>,...}`
prints the name of a compound. If a list of labels is given as argument, a list of names separated by `\compoundseparator` is created. In `implicit` mode this command also creates a new compound name if the label is used for the first time and a custom name has not already been assigned to this compound.
- `\compound*` The starred version works in almost exactly the same way as `\compound`. The only difference is that it does not create any output at all. However, it still creates

the label in `implicit` mode. It can thus be used to create “hidden” compounds in `implicit` mode. This, i.e., can be useful if some compounds are depicted in an illustration or scheme which are only later or even never mentioned in the text but the numbering scheme should take care of them.

2.3 Customisation

The commands in this section can be used to fine-tune the appearance of the compound names. In order to change the default behaviour you have to `\renewcommand` the corresponding commands. The defaults for every command are given in parentheses.

<code>\compoundseparator</code>	<code>\compoundseparator (,\penalty\@m__)</code> defines the separator in a list of compound names.
<code>\compoundglobalprefix</code>	<code>\compoundglobalprefix ()</code> defines the prefix for a list of compound names. This will be printed also in case the list has length one.
<code>\compoundglobalsuffix</code>	<code>\compoundglobalsuffix ()</code> defines the suffix for a list of compound names. This will be printed also in case the list has length one.
<code>\compoundprefix</code>	<code>\compoundprefix ()</code> defines the prefix for every compound.
<code>\compoundsuffix</code>	<code>\compoundsuffix ()</code> defines the suffix for every compound.
<code>\compoundstyle</code>	<code>\compoundstyle (\textbf)</code> defines the style of each name.
<code>\printcompound</code>	<code>\printcompound ({\compoundprefix}{\compoundstyle{#1}}{\compoundsuffix})</code> is used to actually format the name of each compound. If the previous possibilities are not sufficient to meet your formatting demands the thing you should redefine is this one.

2.4 Examples

The following examples using

```
\declarecompound{label1}
\declarecompound{label2}
\declarecompound[5b]{label3}
\compound{label1} and \compound{label1,label2,label3}
```

should clarify the meaning of the above commands. The first two `\declarecommands` could be omitted in `implicit` mode.

- Using the defaults results in **1** and **1, 2, 5b**.
- `\renewcommand{\compoundstyle}{\underbar}`

gives 1 and 1, 2, 5b.

- `\renewcommand{\compoundseparator}{;}`

gives **1** and **1; 2; 5b**.

- `\renewcommand{\compoundglobalprefix}{(}`
`\renewcommand{\compoundglobalsuffix}{)}`

gives **(1)** and **(1, 2, 5b)**.

- `\renewcommand{\compoundprefix}{(}`
`\renewcommand{\compoundsuffix}{)}`

gives **(1)** and **(1), (2), (5b)**.

- `\renewcommand{\compoundglobalprefix}{\textbf{[}}`
`\renewcommand{\compoundglobalsuffix}{\textbf{]}}`
`\renewcommand{\compoundprefix}{\ensuremath{\langle}}`
`\renewcommand{\compoundsuffix}{\ensuremath{\rangle}}`
`\renewcommand{\compoundstyle}{\emph}`

gives **[1]** and **[1], [2], [5b]**.

As shown above customization is not limited solely to parentheses etc. but can include formatting commands, too.

3 The implementation

3.1 Identification

The package identifies itself at the top using something like

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{chemcompounds}
3 [\filedate\space \fileversion\space Dictionary for compound numbering]
```

3.2 Package options

`\ifchemcompounds@implicit` Define a new boolean variable defining whether `implicit` mode is enabled.

```
4 \newif\ifchemcompounds@implicit
```

`implicit` The following package options set `ifchemcompounds@implicit` either to true or
`noimplicit` false. The default is `implicit` mode.

```
5 \DeclareOption{implicit}{\chemcompounds@implicittrue}
6 \DeclareOption{noimplicit}{\chemcompounds@implicitfalse}
7 \ExecuteOptions{implicit}
```

Process options.

```
8 \ProcessOptions
```

3.3 User interface

The work flow for creating and accessing compound names was borrowed from the definition of `\bibitem` and `\cite`.

<code>\compoundseparator</code>	The following definitions define the default layout of the names in the text (no surrounding parentheses, multiple compound names separated by comma, names in bold face). <code>\printcompound</code> defines the way each name is printed.
<code>\compoundglobalprefix</code>	
<code>\compoundglobalsuffix</code>	
<code>\compoundprefix</code>	9 <code>\def\compoundseparator{,\penalty\@m\ }</code>
<code>\compoundsuffix</code>	10 <code>\let\compoundglobalprefix\@empty</code>
<code>\compoundstyle</code>	11 <code>\let\compoundglobalsuffix\@empty</code>
<code>\printcompound</code>	12 <code>\let\compoundprefix\@empty</code> 13 <code>\let\compoundsuffix\@empty</code> 14 <code>\def\compoundstyle{\textbf}</code> 15 <code>\def\printcompound#1{{\compoundprefix}{\compoundstyle{#1}}{\compoundsuffix}}</code>
<code>\declarecompound</code>	This command is used to assign a name to a compound. It just looks ahead whether an optional argument was given and calls the appropriate internal command. To avoid problems with the creation of the labels this command is only allowed in the preamble. 16 <code>\def\declarecompound{\@ifnextchar[\@ldeclarecompound\@declarecompound}</code> 17 <code>\@onlypreamble\declarecompound</code>
<code>\ifchemcompounds@print</code>	Define a new boolean variable indicating whether the starred version of <code>\compound</code> was used. 18 <code>\newif\ifchemcompounds@print</code>
<code>\compound</code>	This command will finally print the name associated with a compound label. The
<code>\compound*</code>	starred version just creates the label (in <code>implicit</code> mode) without printing the value. The command itself just checks whether the starred version is used, sets an internal flag appropriately and calls the internal command <code>\@compound</code> . 19 <code>\DeclareRobustCommand{\compound}{%</code> 20 <code>\@ifstar{\chemcompounds@printfalse\@compound}</code> 21 <code>{\chemcompounds@printtrue\@compound}</code> 22 <code>}</code>

3.4 Internal commands

<code>\@compound</code>	This command retrieves the name associated with a compound and prints it in the text using the previously defined format. The code is a modified version of the definition of <code>\cite</code> in <code>latex.ltx</code> . If this command is invoked by <code>\compound*</code> , <code>\ifchemcompounds@print</code> will be false and all output will be suppressed. 23 <code>\def\@compound#1{%</code> Print optional prefix to a list of compounds. 24 <code>\ifchemcompounds@print</code> 25 <code>\compoundglobalprefix</code> 26 <code>\fi</code> Now loop over every label in the argument list. 27 <code>\let\@compounda\@empty</code> 28 <code>\@for\@compoundb:=#1\do{%</code> 29 <code>\edef\@compoundb{\expandafter\@firstofone\@compoundb}%</code>
-------------------------	--

Print separator. Note that it is empty for the first entry and `\compoundseparator` otherwise.

```
30 \ifchemcompounds@print
31 \@compounda
32 \def\@compounda{\compoundseparator}}%
33 \fi
```

If compound is undefined print '?' and raise a warning.

```
34 \@ifcompoundundefined{\@compoundb}{%
35 \ifchemcompounds@print
36 \mbox{\reset@font\bfseries ?}%
37 \fi
38 \G@refundefinedtrue
39 \@latex@warning
40 {compound '\@compoundb' on page \thepage\space undefined}%
41 }%
```

If compound is known print formatted name.

```
42 \ifchemcompounds@print
43 \mbox{\printcompound{\@nameuse{comp@\@compoundb}}}%
44 \fi
45 }%
```

In implicit mode `\@createcompoundhook` will generate a new name if this has not been done before. In `noimplicit` mode this does nothing.

```
46 \@createcompoundhook{\@compoundb}%
47 }%
```

Print optional suffix to a list of compounds.

```
48 \ifchemcompounds@print
49 \compoundglobalsuffix
```

Although nothing is printed, under certain conditions an additional space is created. Remove it.

```
50 \else
51 \unskip
52 \fi
53 }
```

`\chemcompounds@counter` Define a new counter which will be used for generating the compound names.

```
54 \newcounter{chemcompounds@counter}
```

`\chemcompounds@label` The next command will be used in the `.aux` file and defines a new label for every compound.

```
55 \def\chemcompounds@label{\@newl@bel {comp}}
```

`\chemcompounds@writelabel` Write the label and its value to the `aux` file.

```
56 \def\chemcompounds@writelabel#1#2{%
57 \if@filesw
58 \begingroup
59 \def\protect{\noexpand}%
60 \immediate\write\@auxout{
61 \string\chemcompounds@label{#1}{#2}
62 }%
63 \endgroup
```

```

64 \fi
65 \ignorespaces
66 }

\@ldeclarecompound The next command gets called if an additional argument was supplied to
\declarecompound. It creates the compound with the given name as soon as
the aux file is writeable. This command can only be used in the preamble.
67 \def\@ldeclarecompound[#1]#2{%
68 \AtBeginDocument{\@createcompound[#1]{#2}}
69 }
70 \@onlypreamble\@ldeclarecompound

\@declarecompound In implicit mode this command does nothing since default names are created
automatically by \compound.
71 \ifchemcompounds@implicit
72 \let\@declarecompound\@gobble

In noimplicit mode this simply creates the compound as soon as the aux file is
writeable.
73 \else
74 \def\@declarecompound#1{%
75 \AtBeginDocument{\@createcompound{#1}}
76 }
77 \fi
78 \@onlypreamble\@declarecompound

This command can only be used in the preamble.

\@ifcompoundundefined Test whether a compound has already been defined by testing the associated label.
79 \def\@ifcompoundundefined#1{%
80 \@ifundefined{comp@#1}
81 }

\@createcompound This command is used to create a new compound name. It just looks ahead
whether an optional argument was given and calls the appropriate command.
82 \def\@createcompound{%
83 \@ifnextchar[\@@lcreatecompound\@@createcompound
84 }

\@@createcompound If a compound name has not yet been created this command increments
chemcompounds@counter and takes the new value as the compound name. The
new compound name is written to the aux file and a flag is set to indicate that a
name for this compound has already been created.
85 \def\@@createcompound#1{%
86 \@ifnotcompoundcreated{#1}{%
87 \stepcounter{chemcompounds@counter}%
88 \chemcompounds@writelabel{#1}{\the\value{chemcompounds@counter}}
89 \@compoundcreated{#1}%
90 }%
91 }

\@@lcreatecompound This command creates a compound name from the first parameter and writes it
to the aux file. A flag is set to indicate that a name for this compound has already
been created.

```

```

92 \def\@@lcreatecompound[#1]#2{%
93   \chemcompounds@writelabel{#2}{#1}
94   \@compoundcreated{#2}%
95 }

```

\@compoundcreated Set a flag indicating that the compound name has been created. This is done by defining an appropriate label in `implicit` mode.

```

96 \ifchemcompounds@implicit
97   \def\@compoundcreated#1{%
98     \global\@namedef{compc@#1}{}%
99   }

```

In `noimplicit` mode this is unnecessary thus just gobble the argument.

```

100 \else
101   \let\@compoundcreated\@gobble
102 \fi

```

\@ifnotcompoundcreated Check whether a new compound name has already been created. In `implicit` mode existence of the corresponding label (flag) is checked. If it does not exist the code given as the second argument is executed.

```

103 \ifchemcompounds@implicit
104   \def\@ifnotcompoundcreated#1#2{%
105     \@ifundefined{compc@#1}{#2}{}%
106   }

```

In `noimplicit` mode the label (flag) is unused and a new name will always be created. Therefore just execute the code given as second argument.

```

107 \else
108   \let\@ifnotcompoundcreated\@secondoftwo
109 \fi

```

\@createcompoundhook This command gets called everytime a compound name is printed. In `implicit` mode this command is equal to creating a new compound name.

```

110 \ifchemcompounds@implicit
111   \let\@createcompoundhook\@createcompound

```

In `noimplicit` mode this is unnecessary. Therefore just gobble the argument.

```

112 \else
113   \let\@createcompoundhook\@gobble
114 \fi

```

```

115 \endinput

```

Change History

1.0.0	compound name.	5
General: First version posted on	1.1.0	
CTAN.	1	
1.0.1	\@@createcompound: New.	7
\compound: Replaced \hbox by	\@@lcreatecompound: New.	7
\mbox to work around some	\@compoundcreated: New.	8
spacing issues when printing the	\@createcompound: New.	7
	\@createcompoundhook: New.	8

<code>\@declarecompound:</code> Completely rewritten.	7	<code>\@ifcompoundundefined:</code> Changed label prefix to 'comp'.	7
<code>\@ifcompoundundefined:</code> New. . . .	7	<code>\@ifnotcompoundcreated:</code> Changed label prefix to 'compc'.	8
<code>\@ifnotcompoundcreated:</code> New. . .	8	<code>\chemcompounds@label:</code> Changed label prefix to 'comp' since 'c' is already in use.	6
<code>\@ldeclarecompound:</code> Completely rewritten.	7	<code>\chemcompounds@writelabel:</code> <code>\noexpand</code> the label value. Required for improved robustness.	6
General: Added code to process options.	4	<code>\compound:</code> Changed label prefix to 'comp'.	5
<code>\chemcompounds@label:</code> Labels are now prefixed with 'c' making them hopefully unique.	6		
<code>\chemcompounds@writelabel:</code> New.	6		1.1.2
<code>\compound:</code> Changed definition of <code>\compoundseparator</code> . Inserted <code>\@createcompoundhook</code>	5	<code>\@compound:</code> New. Previous functionality moved to <code>\printcompound</code>	5
<code>\compoundseparator:</code> Changed default value to include trailing space.	5	General: Posted on CTAN on 2005/10/24.	1
<code>\declarecompound:</code> Only in preamble.	5	<code>\chemcompounds@writelabel:</code> Added <code>\ignorespaces</code>	6
<code>\ifchemcompounds@implicit:</code> New.	4	<code>\compound:</code> Only check for starred version and call internal command. Functionality moved to <code>\@compound</code>	5
<code>implicit:</code> New option.	4	<code>\compound*</code> : New.	5
<code>noimplicit:</code> New option.	4	<code>\ifchemcompounds@print:</code> New. . .	5
1.1.1 <code>\@compoundcreated:</code> Changed label prefix to 'compc'.	8	<code>\printcompound:</code> New. Previous <code>\@compound</code> command.	5