

The chemidentifier package, version 1

Aliocha SKRZYPCZAK

Version 1.0.0 — frozen

2026-08-17

This manual describes **version 1**, which is frozen: it is kept unchanged for the documents already written against it, and is loaded with `\usepackage[version=1]{chemidentifier}`. Version 2, the one `\usepackage{chemidentifier}` now loads, has its own manual (`chemidentifier-doc.pdf`): it renames `\herechemid` to `\chemidhere`, leaves a compound with no anchor unlinked rather than pointing at nothing, and adds templated families.

Abstract

`chemidentifier` numbers the compounds of a synthesis-chemistry paper in *the logical order of the synthesis* — the order in which you declare them — rather than the order in which they appear in the text. Every printed number is clickable and jumps to the scheme showing the compound. The package is written in L^AT_EX3 (expl3); like `\ref/\label`, everything settles over **two compilations**.

Contents

1	Installation and loading	2
2	The three commands	2
2.1	<code>\chemid*</code> — declaring	2
2.2	<code>\chemid</code> — using	3
2.3	<code>\herechemid</code> — anchoring	4
3	What happens in the table of contents and bookmarks	4
4	Numbering	4
4.1	Keys	4
4.2	Beyond 26 children	4
4.3	Resetting	5
5	Options	5
6	Multilingual documents	5
7	Two points worth knowing	6
7.1	The raw name derived automatically	6
7.2	Links without a target	6

8	Declaring after using	7
9	Text substitution in .pdf_tex figures (LuaLaTeX only)	7
10	Assumed limitations	8

1 Installation and loading

Copy `chemidentifier.sty` next to your document, or run `make install` to install it into `~/texmf`.

```
\usepackage{hyperref}           % preferably before chemidentifier
\usepackage[version=1]{chemidentifier}
```

`hyperref` is not loaded by the package: it is *detected*, leaving you full control over its options and the loading order. If it is missing, everything is still numbered normally, nothing is clickable, and a warning reports it once.

2 The three commands

Command	Role	Output
<code>\chemid*{key}</code>	declaration	none
<code>\chemid{key}</code>	use	the number, clickable
<code>\herechemid{key}</code>	anchor	none

2.1 `\chemid*` — declaring

```
\chemid*{key}
\chemid*{key}{rich name}
\chemid*{key}{rich name}[raw name]
\chemid*{key}[rich name][raw name] % equivalent variant
```

The declaration prints nothing. **The order of declarations sets the numbering:** place them in the preamble, in the order of your synthesis scheme.

```
\chemid*{benz}           % 1
\chemid*{benz.cl}        % 1a
\chemid*{benz.br}        % 1b
\chemid*{amine}          % 2
```

The main counter advances on every new *parent*. Each parent has its own letter counter, so you can declare children later without breaking the numbering: `1` and `1a,b`, then `2`.

A *rich name* entirely replaces the automatic number:

```
\chemid*{target}{target\textsubscript{final}}[final target]
```

gives `targetfinal`. The second argument, in brackets, is the plain-text version used in PDF bookmarks and metadata, where no formatting is allowed. If you omit it, it is derived automatically from the rich name (see §7.1).

Declaring several keys at once

A comma-separated list of keys declares each of them, auto-numbered in turn — the same convention as `\chemid`:

```
\chemid*{mol1,mol2,mol3}    % three parents: 1, 2, 3
\chemid*{mol1.a,mol1.b}     % two children of mol1: 1a, 1b
```

A custom name cannot accompany a multiple declaration — `\chemid*{a,b}{name}` would be ambiguous: which key would it belong to? This is an error; declare that key on its own instead. A comma *inside* a single key (the result of a malformed key, e.g. one built by a macro) is also rejected.

A child without its parent

Declaring children in bulk without ever declaring their parent on its own also works: the parent is created implicitly (§4.1), with its own number, and `\chemid` of its name alone prints it normally, without error:

```
\chemid*{molz.a,molz.b}     % molz was never declared on its own
...
\chemid{molz}                % -> its number
```

2.2 `\chemid` — using

`\chemid` accepts a comma-separated list of keys and works everywhere: body text, `\caption`, section titles.

Call	Result	Rule
<code>\chemid{benz}</code>	1	plain key
<code>\chemid{benz.cl}</code>	1a	child key
<code>\chemid{benz.cl,benz.br,benz.i}</code>	1a-c	3 contiguous: range
<code>\chemid{benz.cl,benz.br}</code>	1a,b	2 contiguous
<code>\chemid{benz.cl,benz.i}</code>	1a,1c	non-contiguous
<code>\chemid{benz,amine.a}</code>	1 and 2a	different families
<code>\chemid{target}</code>	target _{final}	custom name
<code>\chemid{seriesa,seriesb,seriesc}</code>	4-6	3 consecutive parents: range

The order you give is respected as-is: nothing is sorted. An undeclared key produces a compilation error and prints `??`, like an unresolved `\ref`.

Several plain parents cited in a row contract into a range exactly like several children of the same parent do: `\chemid{seriesa,seriesb}` (only two, below `range-threshold`) still gives 4 and 5, but a gap breaks the range rather than being papered over, e.g. 4 and 6 for `\chemid{seriesa,seriesc}` (`seriesb` not cited in between). A custom name, an undeclared key, or a child key never joins such a run — it simply ends it, exactly as a different family would.

2.3 `\herechemid` — anchoring

`\herechemid{key}` places the target of the link. Put it in the figure that shows the compound, typically near the `\caption`:

```
\begin{figure}
  \centering
  \includegraphics{schema}
  \herechemid{benz}\herechemid{benz.cl}\herechemid{benz.br}
  \caption{Halogenation of \chemid{benz}}
\end{figure}
```

It prints nothing and *never* affects the layout, in horizontal mode as well as in vertical mode (this is checked by the test suite). A click brings the reader to the figure; the PDF format does not allow targeting a sub-part of an image, an anchor there being a point, not an area.

Placing the anchor of the same key twice triggers a warning: the PDF target would be ambiguous. Using a key that was never declared is an error. Finally, a key that is used but never anchored is reported at the end of compilation, like an unresolved `\ref`.

3 What happens in the table of contents and bookmarks

A compound number in a section title must behave differently depending on where that title gets reused.

Context	Behaviour
Body text, caption, displayed title	clickable number
Table of contents, list of figures	number shown, <i>not</i> clickable
PDF bookmarks, metadata	plain text, no formatting

The link is dropped from the table of contents because the entry is *already* a link to the section: two nested links cannot be represented in PDF. The three renderings are produced by the same, fully expandable internal code, which guarantees they cannot diverge.

4 Numbering

4.1 Keys

A key is written `parent` or `parent.child` — a single level of hierarchy; `a.b.c` is an error. Keys are case-sensitive and accept digits, hyphens and underscores (`2nd`, `my_key`, `compound-3`), but no comma: that is the list separator of `\chemid` and `\chemid*`.

Declaring `parent.child` without having declared `parent` creates the latter automatically, with its own number, and reports it in the `.log`. The option `implicit-parent=false` turns this into an error instead.

4.2 Beyond 26 children

Letters keep going past **z**: **aa**, **ab**, ... There is no limit at 26.

4.3 Resetting

`\chemidreset` resets the main counter to zero: the next declaration will start again from 1. This is the only reset offered — no resetting to an arbitrary value, no partial reset. Compounds already declared keep the number they were given.

5 Options

They are given at load time, or at any point with `\chemidsetup{...}`; in the latter case they follow the scope of the current $\text{\TeX}$ group.

Option	Default	Role
<code>list-sep</code>	<code>{, }</code>	between two families
<code>last-sep</code>	<code>{ and }</code>	before the last family
<code>sub-sep</code>	<code>{,}</code>	between letters of the same family
<code>range-sep</code>	<code>{-}</code>	inside a range <code>1a-c</code>
<code>range-threshold</code>	<code>3</code>	size from which a range is contracted
<code>format</code>	(empty)	formatting applied to each identifier, e.g. <code>\textbf</code>
<code>main-style</code>	<code>arabic</code>	<code>arabic</code> , <code>alph</code> , <code>Alph</code> , <code>roman</code> , <code>Roman</code>
<code>sub-style</code>	<code>alph</code>	same, for the letter
<code>prefix</code>	<code>{chemid.}</code>	prefix of PDF anchors
<code>unknown-text</code>	<code>{??}</code>	shown for an undeclared key
<code>purify</code>	<code>true</code>	derive the raw name from the rich name
<code>implicit-parent</code>	<code>true</code>	create a missing parent automatically
<code>strict-anchors</code>	<code>false</code>	see §7.2
<code>auto-lang</code>	<code>true</code>	see §6

The chemistry-journal typographic convention — bold numbers — is obtained as follows:

```
\usepackage[format=\textbf]{chemidentifier}
```

6 Multilingual documents

The separator before the last item is, by default, the English `and` (`last-sep`). To choose a fixed language, without `babel`:

```
\usepackage[last-sep={~et~}]{chemidentifier} % at load time
\chemidsetup{last-sep={~et~}}                % or at any time
```

In a thesis mixing several languages with `babel`, the separator automatically follows the current language — the one that `\selectlanguage` just set — without anything further needed:

```
\usepackage[french,ngerman,spanish,italian,english]{babel}
...
\selectlanguage{french}   \chemid{a,b} % ... et ...
\selectlanguage{ngerman} \chemid{a,b} % ... und ...
```

```
\selectlanguage{spanish} \chemid{a,b} % ... y ...
\selectlanguage{italian} \chemid{a,b} % ... e ...
\selectlanguage{english} \chemid{a,b} % ... and ...
```

The mechanism reads `\language`, which babel updates on every `\selectlanguage`, in a fully expandable way: it therefore works identically in the text, the table of contents and PDF bookmarks. Without babel loaded, or for an unregistered language, the package silently falls back to the `last-sep` option.

Five languages are known by default, with their usual babel variants:

Language	Separator	Recognized babel names
English	and	english, american, british, australian, UKenglish, USenglish
French	et	french, francais, acadian, canadien
German	und	german, ngerman, austrian, naustrian
Spanish	y	spanish, mexican
Italian	e	italian

To add or redefine a language:

```
\chemidaddlanguage{portuguese}{ e }
```

To disable the automatic switch — a fixed separator, whatever the current language — set `auto-lang=false`; `last-sep` then becomes the sole source again, as before this feature existed. Like any option, this can be done locally inside a group:

```
\begingroup
  \chemidsetup{auto-lang=false,last-sep={ or else }}
  \chemid{a,b} % ... or else ...
\endgroup
\chemid{a,b} % the language takes over again
```

7 Two points worth knowing

7.1 The raw name derived automatically

When you give a rich name without its raw counterpart, the latter is derived using `expl3`'s `\text_purify:n`: `H\textsubscript{2}O` becomes `H2O`. The result is good for common cases (subscripts, superscripts, bold, italics) but **is not guaranteed** for nested mathematics. For a non-trivial rich name, provide the raw version yourself:

```
\chemid*{k}{\alpha$-D-glucopyranose}[alpha-D-glucopyranose]
```

7.2 Links without a target

Links rely on `\hyperlink/\hypertarget`, resolved by the PDF reader, and not on the `.aux` file. The trade-off is that at the moment `\chemid` prints a number, the package cannot know whether the corresponding anchor will be placed further on: the link is therefore always written, and the missing anchor is reported at the end of compilation.

If you would rather a compound that is never anchored have no link at all, enable `strict-anchors`: anchors are then remembered in the `.aux`, which requires *two* compilations, like `\ref`.

8 Declaring after using

`\chemid` and `\herechemid` may appear *before* the `\chemid*` that declares the key — useful when you don't want to be forced to declare everything at the top of the document, in particular when the table of contents precedes the text:

```
Compound \chemid{mol1} is mentioned here, before its declaration.
...
\chemid*{mol1}
```

Every declared key is recorded in the `.aux` at the end of compilation (number, letter, rich name, raw name), and read back on the next `\begin{document}` — the same mechanism as `\label/\ref`, with the same trade-off: the first compilation shows ?? for any reference ahead of its declaration, the second resolves it, and the result stays stable afterwards, since the order of declarations never depends on where they are used. A key still unknown at the end of the second pass is genuinely undeclared: the error then keeps showing.

If you have no particular constraint, declaring at the top of the preamble remains the simplest approach: the number is then known as soon as the key is read, with no forward reference to resolve.

9 Text substitution in `.pdf_tex` figures (LuaLaTeX only)

A scheme exported by Inkscape (a `.pdf_tex` + `.pdf` pair) can have its compound labels wired to `\chemid`, the same way `psfrag` once patched text into `.eps` figures — without the `.eps/psfrag` baggage: a `.pdf_tex` is just L^AT_EX text calling `\includegraphics`, so a plain, line-by-line substitution is enough. This requires compiling with `lualatex` (or another engine providing `\directlua`): the substitution is implemented in Lua, reading the file as plain text and handing the result back to T_EX once the placeholders are replaced. The file on disk is never modified, so re-exporting from Inkscape loses nothing.

A plain-T_EX catcode-trick alternative (à la `psfrag`) was considered and rejected: a `.pdf_tex` is genuine T_EX/PGF code, full of `\`, `{`, `}`, `%`, `#`, `_`, `~`... Reading it verbatim and then re-executing it as code requires the same character to carry two contradictory catcodes — inert while captured, active while replayed — for arbitrary content, which is precisely what made `psfrag` fragile. Lua sidesteps this entirely by treating the file as a plain string throughout, and handing the result to T_EX only at the very end, read under T_EX's own, ordinary catcode regime.

Put a plain-text placeholder in the drawing for each compound label (TMP1, TMP2... anything, as long as it does not also occur as a substring elsewhere in the figure text), then:

```
\usepackage{graphicx}           % needed by the .pdf_tex itself
\usepackage[version=1]{chemidentifier}
\chemidsetup{ pdftex-font = \sffamily\small } % optional, once

\chemid*{precursor}
\chemid*{product}

\begin{figure}
  \centering
  \chemidkey{TMP1}{precursor}      % TMP1 -> \chemid{precursor}
  \chemidkey[1.3]{TMP2}{product}   % 1.3x bigger than the rest
  \chemidnote{TMPCOND}{K$_2$CO$_3$, acetone, 70~\textcelsius}
```

```

\chemidscheme[0.8]{figures/scheme.pdf_tex} % scale is optional
\herechemid{precursor}\herechemid{product}
\caption{Synthesis of \chemid{product} from \chemid{precursor}.}
\end{figure}

```

Command	Role
<code>\chemidkey[factor]{motif}{key}</code>	placeholder → the compound's current number, e.g. <code>\chemid{key}</code>
<code>\chemidnote[factor]{motif}{text}</code>	placeholder → any other text, free-form
<code>\chemidscheme[scale]{path}</code>	reads the file, substitutes, typesets; also extends <code>\graphicspath</code>

`\chemidscheme` consumes the pending `\chemidkey`/`\chemidnote` list as it reads the file, so the next figure automatically starts from an empty list — there is no separate "clear" step to remember.

The size of a substituted label is the product of three independent factors, so a figure scaled down still reads fine without retouching every label by hand:

1. `pdftex-font` (`\chemidsetup`) – the base style, set once, independently of the body text (Inkscape otherwise reinjects the document's own family, at whatever size the drawing used, typically too large), e.g. `\chemidsetup{ pdftex-font = \sffamily\fontsize{9}{11}\}`
2. the `[scale]` of `\chemidscheme` – the same number handed to the figure's own `\svgscale`, so labels shrink/grow together with the drawing.
3. the optional `[factor]` of `\chemidkey`/`\chemidnote` – one label singled out, relative to the others in the same figure; defaults to 1.

Under a non-Lua engine, `\chemidscheme` raises a clear error instead of silently doing nothing or mis-rendering. `\chemidkey` takes the compound's *current* number: it does not declare or renumber anything, so keys still need a matching `\chemid*` declared elsewhere.

10 Assumed limitations

- A single level of hierarchy (`parent.child`).
- Lists are neither sorted nor deduplicated: what you write is what gets printed — `\chemid{a,b,a}` prints 1, 2 and 1, without error.
- `cleveref` is not handled (out of scope).
- A key removed from the document keeps resolving to its last known value until the `.aux` file is cleared — the same limitation as `\ref`/`\label`.
- The optional arguments of `\chemid*` are only recognized when *glued* to the key, with no space or line break at all: `\chemid*{k}{rich}[raw]`. Any space before `{` or `[` excludes them from being read; the content that follows is then treated as ordinary text, never absorbed.