

Le paquetage chemidentifier, version 1

Aliocha SKRZYPCZAK

Version 1.0.0 — figée

2026-08-17

Ce manuel décrit la **version 1**, qui est figée : elle est conservée telle quelle pour les documents déjà écrits avec elle, et se charge par `\usepackage[version=1]{chemidentifier}`. La version 2, celle que charge désormais `\usepackage{chemidentifier}`, a son propre manuel (`chemidentifier-doc.fr.pdf`) : elle renomme `\herechemid` en `\chemidhere`, laisse sans lien un composé sans ancre plutôt que de renvoyer dans le vide, et ajoute les familles à gabarit.

Résumé

`chemidentifier` numérote les composés d'un article de chimie de synthèse dans *l'ordre logique de la synthèse* — l'ordre où vous les déclarez — et non dans l'ordre où ils apparaissent dans le texte. Chaque numéro imprimé est cliquable et renvoie au schéma qui montre le composé. Le paquetage est écrit en L^AT_EX3 (expl3) ; comme `\ref/\label`, tout se stabilise en **deux compilations**.

Table des matières

1	Installation et chargement	2
2	Les trois commandes	2
2.1	<code>\chemid*</code> — déclarer	2
2.2	<code>\chemid</code> — utiliser	3
2.3	<code>\herechemid</code> — ancrer	4
3	Ce qui se passe dans la table des matières et les signets	4
4	Numérotation	4
4.1	Clés	4
4.2	Au-delà de 26 enfants	4
4.3	Remise à zéro	5
5	Options	5
6	Document multilingue	5
7	Deux points à connaître	6
7.1	Le nom brut déduit automatiquement	6
7.2	Liens sans cible	7
8	Déclarer après avoir utilisé	7

1 Installation et chargement

Copiez `chemidentifier.sty` à côté de votre document, ou lancez `make install` pour l'installer dans `~/texmf`.

```
\usepackage{hyperref}           % avant chemidentifier, de preference
\usepackage[version=1]{chemidentifier}
```

`hyperref` n'est pas chargé par le paquetage : il est *déecté*, pour vous laisser maîtriser ses options et l'ordre de chargement. S'il manque, tout est numéroté normalement, rien n'est cliquable, et un avertissement le rappelle une fois.

2 Les trois commandes

Commande	Rôle	Affichage
<code>\chemid*{clé}</code>	déclaration	aucun
<code>\chemid{clé}</code>	utilisation	le numéro, cliquable
<code>\herechemid{clé}</code>	ancree	aucun

2.1 `\chemid*` — déclarer

```
\chemid*{clé}
\chemid*{clé}{nom riche}
\chemid*{clé}{nom riche}[nom brut]
\chemid*{clé}[nom riche][nom brut]      % variante equivalente
```

La déclaration n'imprime rien. **L'ordre des déclarations fixe la numérotation** : placez-les dans le préambule, dans l'ordre de votre schéma de synthèse.

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

Le compteur principal avance à chaque nouveau *parent*. Chaque parent possède son propre compteur de lettres, si bien que vous pouvez déclarer les enfants plus tard sans casser la numérotation : **1** et **1a,b**, puis **2**.

Un *nom riche* remplace complètement le numéro automatique :

```
\chemid*{cible}{cible\textsubscript{finale}}[cible finale]
```

donne **cible_{finale}**. Le second argument, entre crochets, est la version en texte brut utilisée dans les signets PDF et les métadonnées, où aucune mise en forme n'est admise. Si vous l'omettez, il est déduit automatiquement du nom riche (voir §7.1).

Déclarer plusieurs clés d'un coup

Une liste de clés séparées par des virgules déclare chacune d'elles, auto-numérotée à son tour — la même convention que `\chemid` :

<code>\chemid*{mol1,mol2,mol3}</code>	% trois parents : 1, 2, 3
<code>\chemid*{mol1.a,mol1.b}</code>	% deux enfants de mol1 : 1a, 1b

Un nom personnalisé ne peut pas accompagner une déclaration multiple — `\chemid*{a,b}{nom}` serait ambigu, à quelle clé appartiendrait-il ? C'est une erreur ; déclarez alors cette clé seule. Une virgule à l'intérieur d'une seule clé (résultat d'une clé mal formée, p. ex. générée par macro) est elle aussi rejetée.

Un enfant sans son parent

Déclarer des enfants en bloc sans jamais déclarer leur parent à part fonctionne aussi : le parent est créé implicitement (§4.1), avec son propre numéro, et `\chemid` de son nom seul l'affiche normalement, sans erreur :

<code>\chemid*{molz.a,molz.b}</code>	% molz n'a jamais ete declare seul
...	
<code>\chemid{molz}</code>	% -> son numero

2.2 `\chemid` — utiliser

`\chemid` accepte une liste de clés séparées par des virgules et fonctionne partout : corps du texte, `\caption`, titres de section.

Appel	Résultat	Règle
<code>\chemid{benz}</code>	1	clé simple
<code>\chemid{benz.cl}</code>	1a	clé enfant
<code>\chemid{benz.cl,benz.br,benz.i}</code>	1a-c	3 contiguës : intervalle
<code>\chemid{benz.cl,benz.br}</code>	1a,b	2 contiguës
<code>\chemid{benz.cl,benz.i}</code>	1a,1c	non contiguës
<code>\chemid{benz,amine.a}</code>	1 et 2a	familles différentes
<code>\chemid{cible}</code>	cible _{finale}	nom personnalisé
<code>\chemid{seriea,serieb,seriec}</code>	4-6	3 parents consécutifs : intervalle

L'ordre que vous donnez est respecté tel quel : rien n'est trié. Une clé non déclarée produit une erreur de compilation et s'imprime ??, comme un `\ref` non résolu.

Plusieurs parents simples cités à la suite se contractent en intervalle exactement comme plusieurs enfants d'un même parent : `\chemid{seriea,serieb}` (seulement deux, sous le `range-threshold`) donne encore 4 et 5, mais un trou casse l'intervalle plutôt que d'être gommé, par exemple 4 et 6 pour `\chemid{seriea,seriec}` (`serieb` non cité entre les deux). Un nom personnalisé, une clé non déclarée ou une clé enfant ne rejoignent jamais une telle suite — ils la terminent simplement, comme le ferait une famille différente.

2.3 `\herechemid` — ancrer

`\herechemid{clé}` pose la cible du lien. Placez-la dans la figure qui montre le composé, typiquement près du `\caption` :

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

Elle n'imprime rien et ne modifie *jamais* la mise en page, en mode horizontal comme en mode vertical (c'est vérifié par la suite de tests). Un clic amène le lecteur sur la figure ; le format PDF ne permet pas de viser une sous-partie d'une image, une ancre y étant un point et non une zone.

Poser deux fois l'ancre d'une même clé déclenche un avertissement : la cible PDF serait ambiguë. Utiliser une clé jamais déclarée est une erreur. Enfin, une clé utilisée mais jamais ancrée est signalée en fin de compilation, comme un `\ref` non résolu.

3 Ce qui se passe dans la table des matières et les signets

Un numéro de composé dans un titre de section doit se comporter différemment selon l'endroit où ce titre est réutilisé.

Contexte	Comportement
Corps du texte, légende, titre affiché	numéro cliquable
Table des matières, liste des figures	numéro affiché, <i>non</i> cliquable
Signets PDF, métadonnées	texte brut, sans mise en forme

Le lien est retiré de la table des matières parce que l'entrée est *déjà* un lien vers la section : deux liens imbriqués ne sont pas représentables en PDF. Les trois rendus sont produits par le même code interne, entièrement expansible, ce qui garantit qu'ils ne peuvent pas diverger.

4 Numérotation

4.1 Clés

Une clé s'écrit `parent` ou `parent.enfant` — un seul niveau de hiérarchie ; `a.b.c` est une erreur. Les clés distinguent la casse et acceptent chiffres, tirets et soulignés (`2eme`, `ma_cle`, `compose-3`), mais pas de virgule : c'est le séparateur de liste de `\chemid` et `\chemid*`.

Déclarer `parent.enfant` sans avoir déclaré `parent` crée ce dernier automatiquement, avec son propre numéro, et le signale dans le `.log`. L'option `implicit-parent=false` en fait une erreur.

4.2 Au-delà de 26 enfants

Les lettres continuent au-delà de `z` : `aa`, `ab`, ... Il n'y a pas de limite à 26.

4.3 Remise à zéro

`\chemidreset` remet le compteur principal à zéro : la prochaine déclaration repartira de 1. C'est le seul reset offert — pas de remise à une valeur arbitraire, pas de reset partiel. Les composés déjà déclarés conservent le numéro qui leur a été attribué.

5 Options

Elles se donnent au chargement, ou à tout moment avec `\chemidsetup{...}` ; dans ce dernier cas elles suivent la portée du groupe $\text{\TeX}$ courant.

Option	Défaut	Rôle
<code>list-sep</code>	<code>{, }</code>	entre deux familles
<code>last-sep</code>	<code>{ and }</code>	avant la dernière famille
<code>sub-sep</code>	<code>{,}</code>	entre lettres d'une même famille
<code>range-sep</code>	<code>{-}</code>	dans un intervalle <code>1a-c</code>
<code>range-threshold</code>	<code>3</code>	taille à partir de laquelle on contracte en intervalle
<code>format</code>	(vide)	mise en forme appliquée à chaque identifiant, p. ex. <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>	idem, pour la lettre
<code>prefix</code>	<code>{chemid.}</code>	préfixe des ancres PDF
<code>unknown-text</code>	<code>{??}</code>	affichage d'une clé non déclarée
<code>purify</code>	<code>true</code>	déduire le nom brut du nom riche
<code>implicit-parent</code>	<code>true</code>	créer un parent manquant automatiquement
<code>strict-anchors</code>	<code>false</code>	voir §7.2
<code>auto-lang</code>	<code>true</code>	voir §6

La convention typographique des revues de chimie — numéros en gras — se demande ainsi :

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

6 Document multilingue

Le séparateur devant le dernier élément est, par défaut, l'anglais `and` (`last-sep`). Pour choisir une langue fixe, sans `babel` :

```
\usepackage[last-sep={~et~}]{chemidentfier} % au chargement
\chemidsetup{last-sep={~et~}}                % ou à tout moment
```

Dans une thèse mêlant plusieurs langues avec `babel`, le séparateur suit automatiquement la langue courante — celle que `\selectlanguage` vient de fixer — sans rien demander de plus :

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

<code>\selectlanguage{ngerman}</code>	<code>\chemid{a,b}</code>	% ... und ...
<code>\selectlanguage{spanish}</code>	<code>\chemid{a,b}</code>	% ... y ...
<code>\selectlanguage{italian}</code>	<code>\chemid{a,b}</code>	% ... e ...
<code>\selectlanguage{english}</code>	<code>\chemid{a,b}</code>	% ... and ...

Le mécanisme lit `\language`, que babel met à jour à chaque `\selectlanguage`, de façon entièrement expansible : il fonctionne donc identiquement dans le texte, la table des matières et les signets PDF. Sans babel chargé, ou pour une langue non enregistrée, le paquetage retombe silencieusement sur l'option `last-sep`.

Cinq langues sont connues par défaut, avec leurs variantes babel usuelles :

Langue	Séparateur	Noms babel reconnus
anglais	and	english, american, british, australian, UKenglish, USenglish
français	et	french, francais, acadian, canadien
allemand	und	german, ngerman, austrian, naustrian
espagnol	y	spanish, mexican
italien	e	italian

Pour ajouter ou redéfinir une langue :

<code>\chemidaddlanguage{portuges}{ e }</code>
--

Pour désactiver la bascule automatique — un séparateur fixe, quelle que soit la langue courante — mettez `auto-lang=false` ; `last-sep` redevient alors la seule source, comme avant l'introduction de cette fonctionnalité. Comme toute option, cela peut se faire localement dans un groupe :

<pre> \begingroup \chemidsetup{auto-lang=false,last-sep={ ou bien }} \chemid{a,b} \endgroup \chemid{a,b} </pre>	<pre> % ... ou bien ... % la langue reprend la main </pre>
---	--

7 Deux points à connaître

7.1 Le nom brut déduit automatiquement

Quand vous donnez un nom riche sans son équivalent brut, celui-ci est déduit avec `\text_purify:n` d'expl3 : `H\textsubscript{2}O` devient `H2O`. Le résultat est bon sur les cas courants (indices, exposants, gras, italique) mais **n'est pas garanti** sur des mathématiques imbriquées. Pour un nom riche non trivial, donnez la version brute vous-même :

<code>\chemid*{k}{\alpha\$-D-glucopyranose}[alpha-D-glucopyranose]</code>

7.2 Liens sans cible

Les liens reposent sur `\hyperlink/\hypertarget`, résolus par le lecteur PDF, et non sur le fichier `.aux`. La contrepartie est qu’au moment où `\chemid` imprime un numéro, le packaging ne peut pas savoir si l’ancree correspondante sera posée plus loin : le lien est donc toujours écrit, et l’absence d’ancree est signalée en fin de compilation.

Si vous préférez qu’un composé jamais ancré n’ait carrément aucun lien, activez `strict-anchors` : les ancrées sont alors mémorisées dans le `.aux`, ce qui demande *deux* compilations, comme `\ref`.

8 Déclarer après avoir utilisé

`\chemid` et `\herechemid` peuvent apparaître *avant* le `\chemid*` qui déclare la clé — utile pour ne pas être contraint de tout déclarer en tête de document, en particulier lorsque la table des matières précède le texte :

```
Le compose \chemid{mol1} est cite ici, avant sa declaration.
...
\chemid*{mol1}
```

Chaque clé déclarée est enregistrée dans le `.aux` à la fin de la compilation (numéro, lettre, nom riche, nom brut), et relue au `\begin{document}` suivant — le même mécanisme que `\label/\ref`, avec la même contrepartie : la première compilation affiche ?? pour toute référence en avance sur sa déclaration, la deuxième la résout, et le résultat reste stable ensuite, puisque l’ordre des déclarations ne dépend jamais de l’endroit où elles sont utilisées. Une clé encore inconnue à la fin de la deuxième passe est réellement non déclarée : l’erreur reste alors affichée.

Si vous n’avez pas de contrainte particulière, déclarer en tête de préambule reste ce qu’il y a de plus simple : le numéro est connu dès la lecture de la clé, sans référence en avant à résoudre.

9 Substitution de texte dans les figures `.pdf_tex` (LuaLaTeX uniquement)

Un schéma exporté par Inkscape (une paire `.pdf_tex` + `.pdf`) peut avoir ses étiquettes de composés reliées à `\chemid`, comme `psfrag` rustinait autrefois du texte dans des figures `.eps` — sans les scories du `.eps/psfrag` : un `.pdf_tex` n’est que du texte L^AT_EX qui appelle `\includegraphics`, donc une simple substitution ligne par ligne suffit. Cela demande de compiler avec `lualatex` (ou tout autre moteur fournissant `\directlua`) : la substitution est faite en Lua, qui lit le fichier comme du texte brut et rend la main à T_EX une fois les marqueurs remplacés. Le fichier sur le disque n’est jamais modifié : réexporter depuis Inkscape ne perd donc rien.

Une alternative en T_EX pur, à base de manipulation de catcodes (à la `psfrag`), a été envisagée puis écartée : un `.pdf_tex` est du vrai code T_EX/PGF, plein de `\`, `{`, `}`, `%`, `#`, `_`, `~`... Le lire tel quel puis le ré-exécuter comme du code demanderait qu’un même caractère porte deux catcodes contradictoires — inerte pendant la lecture, actif pendant l’exécution — pour un contenu arbitraire : exactement ce qui rendait `psfrag` fragile. Lua contourne le problème en traitant le fichier comme une simple chaîne de bout en bout, et ne rend la main à T_EX qu’à la toute fin, lu alors sous le régime de catcodes ordinaire de T_EX.

Placez dans le dessin un marqueur texte brut pour chaque étiquette de composé (TMP1, TMP2...ce que vous voulez, à condition qu'il n'apparaisse pas aussi comme sous-chaîne ailleurs dans le texte de la figure), puis :

```
\usepackage{graphicx}           % necessaire au .pdf_tex lui-meme
\usepackage[version=1]{chemidentfier}
\chemidsetup{ pdftex-font = \sffamily\small }    % optionnel, une fois

\chemid*{precuteur}
\chemid*{produit}

\begin{figure}
  \centering
  \chemidkey{TMP1}{precuteur}           % TMP1 -> \chemid{precuteur}
  \chemidkey[1.3]{TMP2}{produit}       % 1,3x plus gros que le reste
  \chemidnote{TMPCOND}{K$_2$CO$_3$, acetone, 70~\textcelsius}
  \chemidscheme[0.8]{figures/schema.pdf_tex} % echelle optionnelle
  \herechemid{precuteur}\herechemid{produit}
  \caption{Synthese de \chemid{produit} a partir de \chemid{precuteur}.)}
\end{figure}
```

Commande	Rôle
<code>\chemidkey[facteur]{motif}{clé}</code>	marqueur → numéro courant du composé, ex. <code>\chemid{clé}</code>
<code>\chemidnote[facteur]{motif}{texte}</code>	marqueur → tout autre texte, libre
<code>\chemidscheme[échelle]{chemin}</code>	lit le fichier, substitue, compose ; étend aussi <code>\graphicspath</code>

`\chemidscheme` consomme la liste `\chemidkey`/`\chemidnote` en attente au fur et à mesure qu'il lit le fichier : la figure suivante repart donc automatiquement d'une liste vide — pas d'étape « vider » séparée à retenir.

La taille d'une étiquette substituée est le produit de trois facteurs indépendants, pour qu'une figure réduite reste lisible sans retoucher chaque étiquette à la main :

1. `pdftex-font` (`\chemidsetup`) – la police de base, réglée une fois pour toutes, indépendamment du texte du corps (sinon Inkscape réinjecte la famille du document, à la taille du dessin, souvent trop grande), ex. `\chemidsetup{ pdftex-font = \sffamily\fontsize{9}{1}`
2. l'[échelle] de `\chemidscheme` – le même nombre transmis au `\svgscale` de la figure, pour que les étiquettes suivent la taille du dessin.
3. le [facteur] optionnel de `\chemidkey`/`\chemidnote` – une étiquette isolée par rapport aux autres de la même figure ; vaut 1 par défaut.

Sous un moteur non-Lua, `\chemidscheme` déclenche une erreur claire plutôt que de ne rien faire silencieusement ou de mal rendre le schéma. `\chemidkey` prend le numéro *courant* du composé : elle ne déclare ni ne renumérote rien, les clés doivent donc toujours avoir leur `\chemid*` déclaré par ailleurs.

10 Limites assumées

- Un seul niveau de hiérarchie (`parent.enfant`).
- Les listes ne sont pas triées ni dédoublonnées : ce que vous écrivez est ce qui est imprimé — `\chemid{a,b,a}` affiche 1, 2 and 1, sans erreur.

- `cleveref` n'est pas géré (hors périmètre).
- Une clé supprimée du document continue de se résoudre à sa dernière valeur connue tant que le `.aux` n'a pas été effacé — la même limite que `\ref/\label`.
- Les arguments optionnels de `\chemid*` ne sont reconnus que *collés* à la clé, sans le moindre espace ni saut de ligne : `\chemid*{k}{riche}[brut]`. La moindre espace avant `{` ou `[` les exclut de la lecture ; le contenu qui suit est alors traité comme du texte ordinaire, jamais absorbé.