

LilyPond

Le système de gravure musicale

Nouveautés

L'équipe de développement de LilyPond

Ce document recense les modifications et les nouvelles fonctionnalités de LilyPond pour la version 2.25.15 (depuis la 2.24).

Pour connaître la place qu'occupe ce manuel dans la documentation, consultez la page Section "Manuels" dans *Informations générales*.

Si vous ne disposez pas de certains manuels, la documentation complète se trouve sur <https://lilypond.org/>.

Ce document a été placé dans le domaine public ; en France, les auteurs renoncent à tous leurs droits patrimoniaux.

Pour LilyPond version 2.25.15

Note : Chaque nouvelle version de LilyPond peut comporter des changements de syntaxe, ce qui peut exiger de modifier les fichiers que vous avez écrits avec des versions précédentes, de telle sorte qu'ils restent fonctionnels avec les nouvelles versions. Afin de mettre à jour vos fichiers, il est **fortement conseillé** d'utiliser l'utilitaire `convert-ly` distribué avec LilyPond et qui est abordé dans Voir Section "Mise à jour avec `convert-ly`" dans *Utilisation des programmes*. `convert-ly` est capable de réaliser la plupart des modifications de syntaxe automatiquement. Les utilisateurs de Frescobaldi peuvent lancer `convert-ly` directement à partir du menu de Frescobaldi en faisant « Outils > Mettre à jour avec `convert-ly`. . . ». D'autres environnements prenant en charge LilyPond sont susceptibles de fournir un moyen graphique de lancer `convert-ly`.

Modifications majeures de LilyPond

- Les marges sont désormais plus larges, suivant ainsi les mises en pages de nombreux éditeurs, et conformément aux recommandations d'Elaine Gould.

Pour retrouver les mêmes réglages que précédemment, notamment dans le but de conserver la mise en page après mise à jour d'une partition à la version 2.25.15, il suffit d'ajouter le code suivant :

```
\paper {
  top-margin = 5\mm
  bottom-margin = 10\mm
  top-system-spacing.basic-distance = 1
  top-markup-spacing.basic-distance = 0
  left-margin = 10\mm
  right-margin = 10\mm
  inner-margin = 10\mm
  outer-margin = 20\mm
  binding-offset = 0\mm
}
```

- Au lieu de générer des sorties PostScript ou SVG par lui-même, LilyPond peut désormais utiliser la bibliothèque Cairo pour produire ses résultats. Il est ici fait référence au « moteur Cairo » qui peut être activé par l'option `-dbackend-cairo` en ligne de commande. Cette fonctionnalité est opérationnelle pour tous les formats de sortie (PDF, SVG, PNG, PostScript) et apporte vitesse et amélioration du rendu SVG en particulier. Néanmoins, les fonctionnalités des moteurs par défaut ne sont pas encore toutes implémentées. Sont entre autres absents le plan des PDF, l'option `-dembed-source-code` pour le PDF et la propriété `output-attributes` pour le SVG.
- Les distances entre la clef et la métrique ainsi qu'entre la clef et l'armure sont désormais calculées différemment. Par voie de conséquence, on obtient un meilleur espacement pour des clefs imposantes, comme `\clef "GG"`, ou très fines comme `\clef "petrucci-c3"`.

L'image suivante illustre les modifications apportées aux positionnements. Le pourcentage indique la différence d'espacement entre la clef et la métrique, et entre la clef et l'armure.

clef + time sig old	clef + time sig new	clef + key sig old	clef + key sig new
	-1.6%		-1.5%
	+0.5%		+0.5%
	0%		0%
	+17.9%		+9.3%
	-5.6%		-5.2%
	-21.8%		-23.5%

Note that, as before, the widest clef in a staff group determines the horizontal position of all clefs in a system; this means, for example, that a piano score containing a treble and an alto clef doesn't change at all.

If you want to restore the previous default values for whatever reason, add

```
\override Staff.Clef.space-alist.time-signature =
  #'(minimum-space . 3.5)
\override Staff.Clef.space-alist.key-cancellation =
  #'(minimum-space . 3.5)
\override Staff.Clef.space-alist.key-signature =
  #'(minimum-space . 4.2)
```

to your score.

Notes à propos de la compilation des sources et à l'attention des empaqueteurs

Cette section est destinée aux enthousiastes qui compilent LilyPond à partir des sources et aux empaqueteurs qui préparent LilyPond pour les différentes distributions. Si vous ne faites partie d'aucun de ces groupes, vous pouvez aisément passer ce qui suit.

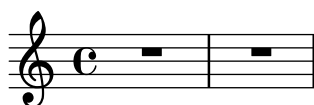
- LilyPond requiert désormais Guile 3.0. Comme précédemment, il est fortement recommandé de générer le *bytecode* des fichiers *.scm* lors de la compilation en lançant `make install-bytecode` avant `make install`.

Nouveautés en matière de notation musicale

Améliorations de la représentation des hauteurs

- Certains faux changements de clefs ont été réglés.

```
{
  R1
  \clef treble
  R1
}
```

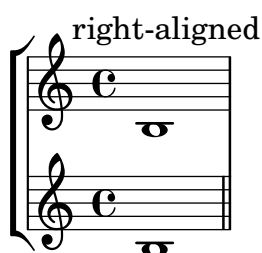
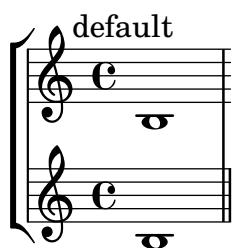


Améliorations en matière de rythme

- Il est désormais possible d'aligner par la droite différents types de barre de mesure.

```
\new StaffGroup
<<
  \new Staff { \textMark "default" b1 }
  \new Staff { b1 \section }
>>

\new StaffGroup
<<
  \new Staff
  { \textMark "right-aligned" b1 }
  \new Staff
  { b1
    \override StaffGroup.BarLine.right-justified = ##t
    \section }
>>
```



- Désormais, les contrôles de mesure (l) créent implicitement des contextes. Les développeurs considèrent que cela n'aura aucun impact sur les partitions courantes. N'hésitez pas à signaler tout problème qui ne trouverait pas de solution de contournement évidente.

- La nouvelle option `span-all-note-heads` permet aux crochets de n-plets d’embrasser toutes les têtes de notes (pas seulement les hampes) comme recommandé par Gould et Ross.



- La subdivision des ligatures automatiques a été retravaillée. Jusqu’à présent, on pouvait constater de nombreuses imperfections dans la manière de subdiviser automatiquement des motifs de ligature complexes en raison de surestimations de la valeur de `baseMoment`. LilyPond est désormais capable de subdiviser correctement la plupart des motifs de ligature sans utiliser la valeur de `baseMoment` pour limiter la subdivision d’une ligature. La simple activation de `subdivideBeams` divise automatiquement tous les intervalles par défaut. Trois nouvelles propriétés ont été introduites pour permettre d’affiner la subdivision automatique des ligatures : `minimumBeamSubdivisionInterval`, `maximumBeamSubdivisionInterval` et `respectIncompleteBeams`. `minimumBeamSubdivisionInterval` limite les intervalles de subdivision de manière identique à ce que `baseMoment` faisait auparavant (réduction de la fréquence des subdivisions de ligatures). `maximumBeamSubdivisionInterval` limite globalement le nombre de tronçons supprimés aux emplacements de subdivision. `respectIncompleteBeams` limite le nombre de moignons lorsque le temps restant ne compléterait pas la métrique de la subdivision. Régler `minimumBeamSubdivisionInterval` à la valeur de `baseMoment` dans tous les cas, y compris lorsque `baseMoment` varie implicitement, préserve le comportement antérieur.
- Sont désormais disponibles des glyphes de crochets « empilés ». Tous les éléments d’un glyphe de crochet ont la même largeur, mais sont verticalement plus compacts. Ces glyphes sont accessibles à l’aide de `\flagStyleStacked` ; un `\flagStyleDefault` permet de retrouver le style de crochet standard.



- Le style de TimeSignature `'single-digit` a été renommé en `'single-number`.

Améliorations en matière d’expressivité

- Les soufflets de style Ferneyhough acceptent désormais un cercle *al niente*.

```
{
  \override Hairpin.circled-tip = ##t
  \override Hairpin.stencil = #flared-hairpin
  b1\< b\> b\> b2 b\< b2 b\!
```

}



- Sont désormais disponibles deux variantes du signe de respiration : `laltcomma` et `raltcomma`. Ces glyphes représentent respectivement les anciens galbes de « `lcomma` » et « `rcomma` » avant leur changement pour un galbe plus courant.

```
{
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.raltcomma" }
  f'2 \breathe f' |
}
```



Améliorations en matière de reprises

- Les fins alternatives d'un `\repeat volta` ne créent plus de barre de mesure invisible. Ceci peut affecter le saut de ligne, l'espacement horizontal et l'extension d'un `VoltaBracket` lorsqu'une alternative débute ou se termine en l'absence de barre de mesure. Dans le cas d'un changement non désiré, il est possible d'ajouter un `\bar ""` ou toute autre commande créant un objet `BarLine` précisément à ce point.
- Grâce à la propriété `printInitialRepeatBar`, il est désormais possible d'afficher automatiquement une barre de reprise même lorsqu'elle intervient en début de pièce.



- Le positionnement du numéro de *volta* relativement au crochet de reprise peut désormais s'ajuster à l'aide de la propriété `volta-number-offset` du `VoltaBracket`.

Améliorations en matière d'annotations éditoriales

- Les objets graphiques `NoteName` sont désormais centrés horizontalement par défaut.

Améliorations en matière de fontes et de mise en forme du texte

- La nouvelle commande de *markup* `\bar-line` permet d'imprimer une barre de mesure dans du texte.

```
\markup {
  \override #'(word-space . 2)
  \line {
    Examples
    \fontsize #-5 \translate-scaled #'(0 . 2) {
      \bar-line ":|."
      \bar-line ".|:"
      \bar-line ";!S!;"
    }
  }
}
```

```

        \bar-line "]"{|}["
      }
    }
  }

```

Exemples 

- La syntaxe permettant de modifier les fontes musicales et textuelles a changé. Au lieu de

```

\paper {
  #(define fonts
    (set-global-fonts
      #:music "Nom de la fonte musicale"
      #:brace "Nom de la fonte musicale d'accolades"
      #:roman "Nom de la fonte à empattements"
      #:sans "Nom de la fonte sans empattements"
      #:typewriter "Nom de la fonte monospace"))
}

```

ou

```

\paper {
  #(define fonts
    (make-pango-font-tree
      "Nom de la fonte à empattements"
      "Nom de la fonte sans empattements"
      "Nom de la fonte monospace"
      factor))
}

```

la syntaxe consacrée est dorénavant

```

\paper {
  property-defaults.fonts.music = "Nom de la fonte musicale"
  property-defaults.fonts.serif = "Nom de la fonte à empattements"
  property-defaults.fonts.sans = "Nom de la fonte sans empattement"
  property-defaults.fonts.typewriter = "Nom de la fonte monospace"
}

```

Contrairement aux anciennes pratiques, la nouvelle syntaxe n'interfère en rien dans la taille des fontes, qui doit se gérer séparément à l'aide de `set-global-staff-size` ou `layout-set-staff-size`.

La liste associative ne comporte pas de clé `brace` ; les glyphes d'accolade sont désormais toujours pris dans la fonte musicale. Il est néanmoins possible d'y déroger en utilisant une famille de fontes supplémentaire, comme dans l'exemple suivant (la fonte `LilyJAZZ` doit alors être disponible) :

```

\layout {
  \context {
    \Score
    \override SystemStartBrace.font.music = "lilyjazz"
  }
}

```

```

\new PianoStaff <<
  \new Staff { c' }
  \new Staff { c' }

```


>>

```
\markup \override #'(fonts . ((music . "lilyjazz"))) \left-brace #20
```

Dans la mesure où `fonts` est une simple propriété, il est possible de lui apporter dérogation sur la base d'un objet graphique, comme par exemple

```
\layout {
  \override Score.SectionLabel.fonts.roman = "Custom font"
}
```

Ceci s'avère préférable à l'utilisation de la propriété `font-name`, cette dernière rendant ineffectives les commandes telles que `\bold` et requérant d'inclure « Bold » dans la chaîne de `font-name`. L'utilisation de `fonts` ne provoque pas de tels effets.

- La commande de *markup* `\lookup` n'est désormais disponible que pour les accolades ; pour les autres glyphes, c'est la commande `\musicglyph` qu'il faut utiliser. Au lieu de `\lookup`, il vaut d'ailleurs mieux lui préférer `\left-brace`.
- Lorsqu'une fonte musicale est utilisée dans un *markup* – typiquement pour une indication de nuance – et qu'un glyphe en était absent, celui-ci était rendu dans une fonte textuelle normale. Ceci n'est plus le cas, et un avertissement est alors émis quant au glyphe manquant. Afin d'utiliser une fonte textuelle, il faut utiliser l'une des commande de *markup* `\serif`, `\sans` ou `\typewriter`, comme ici par exemple.

```
dolceP =
#(make-dynamic-script
#{
  \markup {
    \serif \normal-weight dolce
    p
  }
#})

{ c'\dolceP }
```



- Les petites capitales s'obtiennent désormais en réglant `font-variant` sur `small-caps`, plutôt qu'en fixant `font-shape` à `caps`. Dans la mesure où la raison d'être de `font-shape` est de pouvoir accéder à l'italique, ce changement rend possible l'utilisation conjointe de petites capitales et de l'italique.
- La propriété `font-series` est désormais plus flexible et accepte des valeurs telles que `semibold` et `light` en plus des seules `normal` et `bold`. La valeur `medium` est désormais une valeur intermédiaire entre `normal` et `bold` plutôt qu'un équivalent de `normal`. Par conséquent, la commande de *markup* `\medium` a été renommée en `\normal-weight`.
- La nouvelle propriété `font-stretch` permet de sélectionner une fonte resserrée ou expansée.
- Le texte d'un objet `VoltaBracket` tel que défini par un `\override Score.VoltaBracket.text = ...` ou `\set Score.repeatCommands = ...` n'est plus automatiquement rendu dans une fonte musicale ; il faut pour cela utiliser la commande de *markup* `\volta-number` sur les parties qui le nécessitent. Il faudra donc, par exemple, convertir

```
\set Score.repeatCommands = #'((volta "2, 5"))
```

en

```
\set Score.repeatCommands =
  #`((volta ,#{ \markup {
    \concat { \volta-number 2 , }
    \volta-number 5 }
  #}))
```

- Dans un *markup*, les doigtés (`\markup \finger`) et chiffrages d'accord (`\markup \figured-bass`) sont désormais adaptés à la taille du texte normal en présence d'un `\fontsize`.

```
myText = \markup {
  The fingering \finger { 5-4 } for a \figured-bass { 7 "6\\" } ...
}
```

```
\myText
\markup\fontsize #6 \myText
```

The fingering 5-4 for a 7 6 ...

The fingering 5-4 for a 7 6 ...

Le comportement antérieur peut être retrouvé en activant les variables globales `legacy-figured-bass-markup-fontsize` et `legacy-finger-markup-fontsize`, soit :

```
#{set! legacy-figured-bass-markup-fontsize #t}
#{set! legacy-finger-markup-fontsize #t}
```

```
myText = \markup {
  The fingering \finger { 4-5 } for a \figured-bass { 5+ 6 } ...
}
```

```
\myText
\markup\fontsize #6 \myText
```

The fingering 4-5 for a 5 6 ...

The fingering 4-5 for a 5 6 ...

- Pour plus de clarté, la commande de *markup* `\roman` a été renommée en `\serif`. Aussi, pour modifier une propriété `font-family` réglée à `sans` ou `typewriter`, il faut la définir à `serif`, non plus à `roman`.
- La commande de *markup* `\text` a été supprimée. Doivent être utilisées en remplacement les commandes `\serif`, `\sans` ou `\typewriter`. Si ces commandes permettaient de définir le style de fonte *uniquement lorsqu'une fonte textuelle était utilisée* (non une fonte musicale comme pour les nuances), elles déterminent désormais *à la fois* le style de fonte et l'utilisation d'une fonte textuelle.

Nouveautés en matière de notation spécialisée

Améliorations pour les notations anciennes

- Pour plus de cohérence avec les clefs anciennes, cinq nouvelles clefs mensurales sont disponibles : "mensural-f2", "mensural-f3", "mensural-f4" (identique à "mensural-f"), "mensural-f5", "mensural-g1" et "mensural-g2" (identique à "mensural-g").
- La métrique et le style d'altération à l'armure d'un contexte `PetrucciStaff` sont désormais identiques à ceux d'un `MensuralStaff`.
- Les ligatures mensurales blanches prennent désormais en charge quelques cas rares (semi-brève isolée ou non), et autorisent des affinages permettant d'indiquer quelque hampe non nécessaire.

```

\score {
  \relative {
    \set Score.timing = ##f
    \set Score.measureBarType = #'()
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-c4"
    \[ a1 g f e \]
    \[ a1 g\longa \]
    \[ \once \override NoteHead.left-down-stem = ##t
      a\breve b
      \once \override NoteHead.right-down-stem = ##t
      g\longa \]
    \[ \once \override NoteHead.right-down-stem = ##t
      b\maxima
      \once \override NoteHead.right-up-stem = ##t
      g\longa \]
  }
  \layout {
    \context {
      \Voice
      \remove Ligature_bracket_engraver
      \consists Mensural_ligature_engraver
    }
  }
}

```



- L'utilisation de `gregorian.ly` est désuète. Bien que toujours fonctionnelle pour raison de compatibilité ascendante, elle devrait être remplacée par l'utilisation d'un contexte `VaticanaScore`, avec si besoin des adaptations manuelles dans le bloc `\layout`. Un code tel que

```

\include "gregorian.ly"

\score {
  \new VaticanaStaff { ... }
}

```

devrait devenir

```
\new VaticanaScore {  
  \new VaticanaStaff { ... }  
}
```

```
\layout {  
  indent = 0  
  ragged-last = ##t  
}
```

Autres améliorations diverses

- L'inclusion d'images PNG est désormais possible à l'aide de la commande de `markup \image`. Ceci vient en supplément de la commande `\epsfile` pour les images EPS.

La commande `\image` traite aussi bien les images PNG que EPS, à ceci près que la commande `\image` insère un fond blanc, contrairement à `\epsfile`.

- La nouvelle commande de `markup \qr-code` permet d'insérer un QR-code de la taille spécifiée pour l'URL correspondante. Ceci peut servir à fournir un lien vers le site du compositeur ou de l'éditeur, ou bien vers les sources LilyPond ou des enregistrements, etc.

```
\markup \qr-code #10 "https://lilypond.org"
```



- Ont été ajoutés à la fonte Emmentaler les glyphes figure-dash (U+2012) et en-dash (U+2013) (quart de catradin).
- Une espace pour nombres (U+2007), une espace fine (U+2009) et une espace ultrafine (U+200A) ont été ajoutées à la fonte Emmentaler.
- L'option `-dinclude-settings` peut désormais apparaître à plusieurs reprises, afin de pouvoir inclure différentes feuilles de style.
- Dans l'utilisation conjointe de \LaTeX et `lilypond-book`, les images au fil du texte sont désormais décalées verticalement. Ce décalage peut se contrôler en ligne de commande à l'aide de l'option `--inline-vshift` et, localement, en ajoutant `inline` en argument aux options de l'extrait.
- Deux nouvelles options en ligne de commande font leur apparition : `-dfirst` and `-dlast`. Elles sont équivalents aux réglages respectifs de `showFirstLength` et `showLastLength` dans un fichier LilyPond. Par exemple, taper

```
lilypond -dlast=R1*5 ...
```

aura pour résultat que LilyPond générera seulement les cinq dernières mesure (partant du principe d'une métrique à 4/4).

- Est désormais disponible un nouveau manuel constituant un index de tous les objets graphiques (*grobs*) de LilyPond. Il est basé sur le travail de Joram Berger pour LilyPond 2.19 (<https://github.com/joram-berger/visualindex>).
- L'impression des arpeggios a été améliorée de par l'utilisation de nouvelles valeurs par défaut différentes pour la propriété `Arpeggio.positions`. Les ajustements à cette propriété sont susceptibles d'évoluer.
- LilyPond prend en charge les annotations au fil du texte, autrement dit des sortes de notes de bas de page entre les systèmes. Bien que ce ne soit pas nouveau, puisque disponible depuis la version 2.15.17 publiée en 2011, cette possibilité n'était pas si stable et manquait de documentation jusqu'à présent.
- Le script `lilysong` a été supprimé. En dehors du fait qu'il ne disposait d'aucune documentation, il n'était plus maintenu depuis fort longtemps. De plus, il reposait sur un programme externe de synthèse vocale – festival – laissé à l'abandon.
- Deux nouveaux styles sont disponibles pour la propriété d'objet graphique `space-alist` : `shrink-space` et `semi-shrink-space`. Ils contractent les espaces au lieu de les étendre. Ils sont aussi directement utilisés par LilyPond afin d'améliorer le formatage des portées resserrées.

- Le binaire lilypond dispose d'une nouvelle option en ligne de commande, `-dstaff-size`, pour régler la taille globale des portées. Elle est équivalente à la présence d'un `set-global-staff-size` dans un fichier LilyPond.
- En remplacement des fonctions `\bookOutputName` et `\bookOutputSuffix`, nous recommandons dorénavant l'utilisation des variables de papier `output-filename` et `output-suffix`. Bien que les premières restent pleinement fonctionnelles, ces dernières sont plus cohérentes et facilement compréhensibles, notamment si elles sont combinées avec des variables de papier prédéfinies.
- La propriété `Stem.details.lengths` accepte maintenant des paires en tant qu'éléments de liste. Ceci permet de définir séparément les longueurs de hampe ascendante ou descendante.
- La fonction `ly:self-alignment-interface::aligned-on-x-parent` – utilisée par de nombreux objets graphiques pour calculer leur x-offset – écoute désormais la nouvelle propriété `X-alignment-extent` de `PaperColumn`. Activée par défaut, elle fournit une largeur de secours au *grob* `PaperColumn` dans le cas où il ne contiendrait pas de tête de note. Ceci permet d'aider à l'alignement des scripts de nuance attachés à des silences invisibles, entre autres.

```

music =
  \new Staff <<
    { f'2 g'2 }
    { s4\f s\f s\f s\f }
  >>

\score {
  \music
}

\score {
  \music

  \layout {
    \context {
      \Score
      \override PaperColumn.X-alignment-extent = ##f
    }
  }
}

```



- Les objets `BassFigureContinuation` prennent désormais en charge la `horizontal-line-spanner-interface`. La propriété `padding` a été remplacée par des sous-propriétés correspondantes dans `bound-details`.
- La commande de *markup* `\align-on-other` accepte désormais la valeur `#f` pour l'alignement, indiquant le point de référence d'un *markup*.

- Une nouvelle fonction `\withRelativeDir` est désormais disponible pour les commandes de *markup* qui incluent des fichiers lorsque ces fichiers devraient se trouver relativement au fichier source. Par exemple :

```
\markup { \image #X #3 \withRelativeDir "test.png" }
```

- Le positionnement des crochets horizontaux d'analyse a été amélioré. En particulier, l'objet `HorizontalBracket` a désormais une valeur de `outside-staff-priority` fixée à 800. Par conséquent, il se pourrait que des crochets imbriqués voient leur positionnement modifié. Ceci peut se corriger en ajustant les valeurs de `outside-staff-priority` par un `\tweak`, tout en sachant que le crochet externe doit garder une valeur de priorité supérieure.
- La nouvelle fonction Scheme `to-staff-space` permet de convertir des dimensions absolues (exprimées en diverses unités) en unités d'espace de portée. Par exemple :

```
top-markup-spacing.basic-distance = #(to-staff-space 2 'cm)
```

```
% l'unité par défaut est le point (pt)
```

```
\markup
  \override #`(baseline-skip . ,(to-staff-space 20))
  \column {
    foo
    bar
  }
```

- Les deux nouvelles fonctions de *markup* `\abs-hspace` et `\abs-vspace` permettent de déterminer des dimensions absolues qui subsistent quelle que soit la taille de portée en cours.
- Les données émises par l'option en ligne de commande `-dshow-available-fonts` sont désormais envoyées sur la sortie standard.
- La fonction `ly:font-config-display-fonts` accepte désormais un argument optionnel pour sélectionner le port de sortie.
- La gestion des options Scheme en ligne de commande gagne en robustesse. Quelques changements mineurs ont été rendus nécessaires par cette nouvelle implémentation.
 - Sur la ligne de commande, l'argument à l'option `-dpaper-size` ne nécessite plus d'être mis entre guillemets. En d'autres termes, mentionner `-dpaper-size=a3` est tout à fait valide.
 - L'option `pixmap-format` requiert désormais une valeur sous forme de chaîne, non plus un symbole. Rien ne change pour la ligne de commande, mais un appel tel que


```
 #(ly:set-option 'pixmap-format 'pngalpha)
```

 devra être changé en


```
 #(ly:set-option 'pixmap-format "pngalpha")
```

 Il en va de même pour les options `separate-page-formats` et `tall-page-formats`. Notez que `convert-ly` se charge d'effectuer la modification automatiquement.