

New features in 2.16 since 2.14

- Use of `\tempo` specifications in `\midi` blocks (removed in 2.9.16 in favor of explicit `tempoWholesPerMinute` settings) has seen a revival: now any kind of property-setting music is turned into context definitions within of output specifications, allowing for declarations like

```
\layout { \accidentalStyle modern }  
\midi { \tempo 4. = 66 }
```

- The LilyPond G clef has been redesigned - upper loop is now more balanced, bottom crook sticks out less and the "spine" (main vertical line) is more evenly curved. The old and new versions can be compared by looking at the documentation: [old version](#), [new version](#).
- Lilypond's stencil commands have been simplified to allow for less code duplication and better height approximations of graphical objects. The following stencil commands have been eliminated:
 - `beam`
 - `bezier-sandwich`
 - `bracket`
 - `dashed-slur`
 - `dot`
 - `oval`
 - `repeat-slash`
 - `zigzag-line`
- Flags are now treated as separate objects rather than as stem parts.



- Two alternative methods for bar numbering can be set, especially for when using repeated music;





- The following is a fundamental change in LilyPond’s music representation: Rhythmic events like `LyricEvent` and `NoteEvent` are no longer wrapped in `EventChord` unless they have been actually entered as part of a chord in the input. If you manipulate music expressions in Scheme, the new behavior may require changes in your code. Calling the music function `\eventChords` or the Scheme function `event-chord-wrap!` converts to the old representation; using one of those might be easiest for keeping legacy code operative.

The advantages of making input and music match more closely are numerous: music functions previously worked differently when used inside or outside of chords. Now they are the same, including all the possibilities of argument parsing. You can now use music variables inside of chords: a construct like

```
tonic=fis'
{ <\tonic \transpose c g \tonic> }
```



would have been unthinkable previously. You can use `#{...#}` for constructing chord constituents. Music functions inside of chords are no longer specially treated and thus accept the same arguments as outside of chords. `\tweak` now works on single notes without needing to wrap them in a chord. In theory, it can also work on command events and lyrics now. Since that was not possible before, it depends on luck on a case-by-case basis whether the tweak internals are already receiving the necessary information. Users are asked to report those cases where they find `\tweak` not working according to reasonable expectations.

- As one consequence, it was possible to reimplement the repetitive chord entry aid `q`. Repeated chords are now replaced right before interpreting a music expression. In case the user wants to retain some events of the original chord, he can run the repeat chord replacement function `\chordRepeats` manually.
- Scheme expressions inside of embedded Lilypond (`#{...#}`) are now executed in lexical closure of the surrounding Scheme code. `$` is no longer special in embedded Lilypond. It can be used unconditionally in Lilypond code for immediate evaluation, similar to how `ly:export` could previously be used. `ly:export` has been removed. As a consequence, `#` is now free to delay evaluation of its argument until the parser actually reduces the containing expression, greatly reducing the potential for premature evaluation.
- Support for jazz-like chords has been improved: Lydian and altered chords are recognised; separators between chord modifiers are now treated independently of separators between “slash” chords and their bass notes (and by default, slashes are now only used for the latter type of separator); additional pitches are no longer prefixed with “add” by default; and the “m” in minor chords can be customized. [Section “Customizing chord names” in *Notation Reference*](#) for more information.
- The `\markuplines` command has been renamed to `\markuplist` for a better match with its semantics and general Lilypond nomenclature.

- The interface for specifying string tunings in tablature has been simplified considerably and employs the scheme function `\stringTuning` for most purposes.
- Beams can now have their slopes preserved over line breaks.



To do this, several callback functions are now deprecated.

- `ly:beam::calc-least-squares-positions`
- `ly:beam::slope-damping`
- `ly:beam::shift-region-to-valid`

Furthermore, `ly:beam::quanting` now takes an additional argument to help calculations over line breaks. All of these functions are now automatically called when setting the `positions` parameter.

- In function arguments music, markups and Scheme expressions (as well as several other syntactic entities) have become mostly interchangeable and are told apart only by evaluating the respective predicate. In several cases, the predicate is consulted by the parser, like when deciding whether to interpret `-3` as a number or a fingering event.
- Music functions (and their close relatives) can now be defined with optional arguments.
- For defining commands executed only for their side-effects, `define-void-function` is now available.
- There is a new `define-event-function` command in analogy to `define-music-function` that can be used for defining music functions acting as post events without requiring a direction specifier (`-`, `^`, or `_`) placed before them.

```
dyn=#(define-event-function (parser location arg) (markup?)
      (make-dynamic-script arg))
\relative c' { c\dyn pfsss }
```



- A list of ASCII aliases for special characters can be included.

```
\paper {
  #(include-special-characters)
}
\markup "&bull; &dagger; &copyright; &OE; &ss; &para;"
```

• † © Æ ß ¶

- There is a new **define-scheme-function** command in analogy to **define-music-function** that can be used to define functions evaluating to Scheme expressions while accepting arguments in Lilypond syntax.
- The construct **#{ ... #}** can now be used not just for constructing sequential music lists, but also for pitches (distinguished from single note events by the absence of a duration or other information that can't be part of a pitch), single music events, void music expressions, post events, markups (mostly freeing users from having to use the **markup** macro), markup lists, number expressions, context definitions and modifications, and a few other things. If it encloses nothing or only a single music event, it no longer returns a sequential music list but rather a void music expression or just the music event itself, respectively.
- Pitches can be used on the right side of assignments. They are distinguished from single note events by the absence of a duration or other information that can't be part of a pitch.
- New command-line option **'--loglevel=level'** to control how much output LilyPond creates. Possible values are ERROR, WARN, BASIC_PROGRESS, PROGRESS, DEBUG.
- **\once \set** now correctly resets the property value to the previous value.



- The alignment of dynamic spanners (hairpins, text crescendo, etc.) is now automatically broken if a different direction is explicitly given.



- Appoggiaturas and acciaccaturas now also work inside a slur, not only inside a phrasing slur. Also, a function **\slashedGrace** was added that does not use a slur from the acciaccatura note.



- To suppress the line on a crescendo text spanner (and other similar spanners), LilyPond now fully supports the **#'style = #'none** property.



- LilyPond.app now supports MacOS X 10.7, thanks Christian Hitz!
- Glissandi can now span multiple lines.