LilyPond

The music typesetter

Internals Reference

The LilyPond development team

For LilyPond version 2.21.0

Table of Contents

This is the Internals Reference (IR) for version 2.21.0 of LilyPond, the GNU music typesetter.

1 Music definitions

1.1 Music expressions

1.1.1 AbsoluteDynamicEvent

Create a dynamic mark.

Syntax: *note*`\x`, where `\x` is a dynamic mark like `\ppp` or `\sfz`. A complete list is in file `ly/dynamic-scripts-init.ly`.

Event classes: Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.20 [dynamic-event], page 43, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320, and Section 2.2.34 [Dynamic_performer], page 321.

Properties:

`name` (symbol):

```
'AbsoluteDynamicEvent
```

Name of this music object.

`types` (list):

```
'(post-event
   event
   dynamic-event
   absolute-dynamic-event)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.2 AlternativeEvent

Create an alternative event.

Event classes: Section 1.2.2 [alternative-event], page 41, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.8 [Bar_number_engraver], page 310.

Properties:

`name` (symbol):

```
'AlternativeEvent
```

Name of this music object.

`types` (list):

```
'(event alternative-event)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.3 AnnotateOutputEvent

Print an annotation of an output element.

Event classes: Section 1.2.3 [annotate-output-event], page 41, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.6 [Balloon_engraver], page 310.

Properties:

`name` (symbol):

```
'AnnotateOutputEvent
```

Name of this music object.

`types` (list):
`'(event annotate-output-event post-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.4 ApplyContext

Call the argument with the current context during interpreting phase.

Properties:

`iterator-ctor` (procedure):
`ly:apply-context-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'ApplyContext`
Name of this music object.

`types` (list):
`'(apply-context)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.5 ApplyOutputEvent

Call the argument with all current grobs during interpreting phase.

Syntax: `\applyOutput #'`*`context func`*

Arguments to *func* are 1. the grob, 2. the originating context, and 3. the context where *func* is called.

Event classes: Section 1.2.4 [apply-output-event], page 42, Section 1.2.31 [layout-instruction-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.82 [Output_property_engraver], page 338.

Properties:

`name` (symbol):
`'ApplyOutputEvent`
Name of this music object.

`types` (list):
`'(event apply-output-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.6 ArpeggioEvent

Make an arpeggio on this note.

Syntax: *note*`-\arpeggio`

Event classes: Section 1.2.5 [arpeggio-event], page 42, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.3 [Arpeggio_engraver], page 308.

Properties:

`name` (symbol):
`'ArpeggioEvent`

Name of this music object.

`types` (list):
`'(post-event arpeggio-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.7 ArticulationEvent

Add an articulation marking to a note.

Syntax: *note*`xy`, where `x` is a direction (`^` for up or `_` for down), or LilyPond's choice (no direction specified), and where `y` is an articulation (such as `-.`, `->`, `\tenuto`, `\downbow`). See the Notation Reference for details.

Event classes: Section 1.2.6 [articulation-event], page 42, Section 1.2.40 [music-event], page 45, Section 1.2.54 [script-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.79 [Note_performer], page 337, and Section 2.2.101 [Script_engraver], page 344.

Properties:

`name` (symbol):
`'ArticulationEvent`
Name of this music object.

`types` (list):
```
'(post-event
   event
   articulation-event
   script-event)
```
The types of this music object; determines by what engraver this music expression is processed.

1.1.8 AutoChangeMusic

Used for making voices that switch between piano staves automatically.

Properties:

`iterator-ctor` (procedure):
`ly:auto-change-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'AutoChangeMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(music-wrapper-music auto-change-instruction)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.9 BarCheck

Check whether this music coincides with the start of the measure.

Properties:

`iterator-ctor` (procedure):
`ly:bar-check-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'BarCheck`

Name of this music object.

`types` (list):
`'(bar-check)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.10 BassFigureEvent

Print a bass-figure text.

Event classes: Section 1.2.7 [bass-figure-event], page 42, Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.37 [Figured_bass_engraver], page 322.

Properties:

`name` (symbol):
`'BassFigureEvent`

Name of this music object.

`types` (list):
`'(event rhythmic-event bass-figure-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.11 BeamEvent

Start or stop a beam.

Syntax for manual control: `c8-[ c c-] c8`

Event classes: Section 1.2.8 [beam-event], page 42, Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.10 [Beam_engraver], page 312, Section 2.2.11 [Beam_performer], page 312, and Section 2.2.47 [Grace_beam_engraver], page 326.

Properties:

`name` (symbol):
`'BeamEvent`

Name of this music object.

`types` (list):
`'(post-event event beam-event span-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.12 BeamForbidEvent

Specify that a note may not auto-beamed.

Event classes: Section 1.2.9 [beam-forbid-event], page 42, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.4 [Auto_beam_engraver], page 308, and Section 2.2.46 [Grace_auto_beam_engraver], page 325.

Properties:

`name` (symbol):
`'BeamForbidEvent`

Name of this music object.

`types` (list):
`'(post-event event beam-forbid-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.13 BendAfterEvent

A drop/fall/doit jazz articulation.

Event classes: Section 1.2.10 [bend-after-event], page 42, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.12 [Bend_engraver], page 312.

Properties:

`name` (symbol):
`'BendAfterEvent`

Name of this music object.

`types` (list):
`'(post-event bend-after-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.14 BreakDynamicSpanEvent

End an alignment spanner for dynamics here.

Event classes: Section 1.2.11 [break-dynamic-span-event], page 42, Section 1.2.13 [break-span-event], page 43, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320.

Properties:

`name` (symbol):
`'BreakDynamicSpanEvent`

Name of this music object.

`types` (list):

```
'(post-event
  break-span-event
  break-dynamic-span-event
  event)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.15 BreathingEvent

Create a 'breath mark' or 'comma'.

Syntax: *note*`\breathe`

Event classes: Section 1.2.14 [breathing-event], page 43, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.14 [Breathing_sign_engraver], page 313, and Section 2.2.79 [Note_performer], page 337.

Properties:

`midi-length` (procedure):

```
breathe::midi-length
```

Function to determine how long to play a note in MIDI. It should take a moment (the written length of the note) and a context, and return a moment (the length to play the note).

`name` (symbol):

```
'BreathingEvent
```

Name of this music object.

`types` (list):

```
'(event breathing-event)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.16 ClusterNoteEvent

A note that is part of a cluster.

Event classes: Section 1.2.15 [cluster-note-event], page 43, Section 1.2.37 [melodic-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.18 [Cluster_spanner_engraver], page 315.

Properties:

`iterator-ctor` (procedure):

```
ly:rhythmic-music-iterator::constructor
```

Function to construct a `music-event-iterator` object for this music.

`name` (symbol):

```
'ClusterNoteEvent
```

Name of this music object.

`types` (list):

```
'(cluster-note-event
  melodic-event
  rhythmic-event
```

`event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.17 CompletizeExtenderEvent

Used internally to signal the end of a lyrics block to ensure extenders are completed correctly when a `Lyrics` context ends before its associated `Voice` context.

Event classes: Section 1.2.16 [completize-extender-event], page 43, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.36 [Extender_engraver], page 322.

Properties:

`name` (symbol):
`'CompletizeExtenderEvent`

Name of this music object.

`types` (list):
`'(completize-extender-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.18 ContextChange

Change staves in Piano staff.

Syntax: `\change Staff =` *`new-id`*

Properties:

`iterator-ctor` (procedure):
`ly:change-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'ContextChange`

Name of this music object.

`types` (list):
`'(translator-change-instruction)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.19 ContextSpeccedMusic

Interpret the argument music within a specific context.

Properties:

`iterator-ctor` (procedure):
`ly:context-specced-music-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):

```
'ContextSpeccedMusic
```

Name of this music object.

`start-callback` (procedure):

```
ly:music-wrapper::start-callback
```

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):

```
'(context-specification music-wrapper-music)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.20 CrescendoEvent

Begin or end a crescendo.

Syntax: *note*`\<` ... *note*`\!`

An alternative syntax is *note*`\cr` ... *note*`\endcr`.

Event classes: Section 1.2.17 [crescendo-event], page 43, Section 1.2.40 [music-event], page 45, Section 1.2.61 [span-dynamic-event], page 48, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320, and Section 2.2.34 [Dynamic_performer], page 321.

Properties:

`name` (symbol):

```
'CrescendoEvent
```

Name of this music object.

`types` (list):

```
'(post-event
   span-event
   span-dynamic-event
   crescendo-event
   event)
```

The types of this music object; determines by what engraver this music expression is processed.

1.1.21 DecrescendoEvent

Begin or end a decrescendo.

Syntax: *note*`\>` ... *note*`\!`

An alternative syntax is *note*`\decr` ... *note*`\enddecr`.

Event classes: Section 1.2.18 [decrescendo-event], page 43, Section 1.2.40 [music-event], page 45, Section 1.2.61 [span-dynamic-event], page 48, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320, and Section 2.2.34 [Dynamic_performer], page 321.

Properties:

`name` (symbol):

```
'DecrescendoEvent
```

Name of this music object.

`types` (list):
```
'(post-event
  span-event
  span-dynamic-event
  decrescendo-event
  event)
```
The types of this music object; determines by what engraver this music expression is processed.

1.1.22 DoublePercentEvent

Used internally to signal double percent repeats.

Event classes: Section 1.2.19 [double-percent-event], page 43, Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.29 [Double_percent_repeat_engraver], page 319.

Properties:

`name` (symbol):
```
'DoublePercentEvent
```
Name of this music object.

`types` (list):
```
'(event double-percent-event rhythmic-event)
```
The types of this music object; determines by what engraver this music expression is processed.

1.1.23 EpisemaEvent

Begin or end an episema.

Event classes: Section 1.2.21 [episema-event], page 43, Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.35 [Episema_engraver], page 321.

Properties:

`name` (symbol):
```
'EpisemaEvent
```
Name of this music object.

`types` (list):
```
'(post-event span-event event episema-event)
```
The types of this music object; determines by what engraver this music expression is processed.

1.1.24 Event

Atomic music event.

Properties:

`name` (symbol):
```
'Event
```
Name of this music object.

`types` (list):
`'(event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.25 EventChord

Explicitly entered chords.

When iterated, `elements` are converted to events at the current timestep, followed by any `articulations`. Per-chord postevents attached by the parser just follow any rhythmic events in `elements` instead of utilizing `articulations`.

An unexpanded chord repetition '`q`' is recognizable by having its duration stored in `duration`.

Properties:

`iterator-ctor` (procedure):
`ly:event-chord-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-sequence::event-chord-length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'EventChord`

Name of this music object.

`to-relative-callback` (procedure):
`ly:music-sequence::event-chord-relative-callback`

How to transform a piece of music to relative pitches.

`types` (list):
`'(event-chord simultaneous-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.26 ExtenderEvent

Extend lyrics.

Event classes: Section 1.2.22 [extender-event], page 43, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.36 [Extender_engraver], page 322.

Properties:

`name` (symbol):
`'ExtenderEvent`

Name of this music object.

`types` (list):
`'(post-event extender-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.27 FingeringEvent

Specify what finger to use for this note.

Event classes: Section 1.2.23 [fingering-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.40 [Fingering_engraver], page 323, Section 2.2.44 [Fretboard_engraver], page 324, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

Properties:

`name` (symbol):
`'FingeringEvent`

Name of this music object.

`types` (list):
`'(post-event fingering-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.28 FootnoteEvent

Footnote a grob.

Event classes: Section 1.2.24 [footnote-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Not accepted by any engraver or performer.

Properties:

`name` (symbol):
`'FootnoteEvent`

Name of this music object.

`types` (list):
`'(event footnote-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.29 GlissandoEvent

Start a glissando on this note.

Event classes: Section 1.2.25 [glissando-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.45 [Glissando_engraver], page 325.

Properties:

`name` (symbol):
`'GlissandoEvent`

Name of this music object.

`types` (list):
`'(post-event glissando-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.30 GraceMusic

Interpret the argument as grace notes.

Properties:

`iterator-ctor` (procedure):
`ly:grace-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length` (moment):
`#<Mom 0>`
The duration of this music.

`name` (symbol):
`'GraceMusic`
Name of this music object.

`start-callback` (procedure):
`ly:grace-music::start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(grace-music music-wrapper-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.31 HarmonicEvent

Mark a note as harmonic.

Event classes: Section 1.2.26 [harmonic-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Not accepted by any engraver or performer.

Properties:

`name` (symbol):
`'HarmonicEvent`
Name of this music object.

`types` (list):
`'(post-event event harmonic-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.32 HyphenEvent

A hyphen between lyric syllables.

Event classes: Section 1.2.27 [hyphen-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.54 [Hyphen_engraver], page 328.

Properties:

`name` (symbol):
`'HyphenEvent`
Name of this music object.

`types` (list):
`'(post-event hyphen-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.33 KeyChangeEvent

Change the key signature.

Syntax: `\key` *name scale*

Event classes: Section 1.2.28 [key-change-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.58 [Key_engraver], page 329, and Section 2.2.59 [Key_performer], page 330.

Properties:

`name` (symbol):
`'KeyChangeEvent`

Name of this music object.

`to-relative-callback` (procedure):
`#<procedure #f (x p)>`

How to transform a piece of music to relative pitches.

`types` (list):
`'(key-change-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.34 LabelEvent

Place a bookmarking label.

Event classes: Section 1.2.29 [label-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.84 [Paper_column_engraver], page 338.

Properties:

`name` (symbol):
`'LabelEvent`

Name of this music object.

`types` (list):
`'(label-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.35 LaissezVibrerEvent

Don't damp this chord.

Syntax: *note*`\laissezVibrer`

Event classes: Section 1.2.30 [laissez-vibrer-event], page 44, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.61 [Laissez_vibrer_engraver], page 330.

Properties:

`name` (symbol):
`'LaissezVibrerEvent`
Name of this music object.

`types` (list):
`'(post-event event laissez-vibrer-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.36 LigatureEvent

Start or end a ligature.

Event classes: Section 1.2.32 [ligature-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.60 [Kievan_ligature_engraver], page 330, Section 2.2.63 [Ligature_bracket_engraver], page 331, Section 2.2.70 [Mensural_ligature_engraver], page 333, and Section 2.2.134 [Vaticana_ligature_engraver], page 354.

Properties:

`name` (symbol):
`'LigatureEvent`
Name of this music object.

`types` (list):
`'(span-event ligature-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.37 LineBreakEvent

Allow, forbid or force a line break.

Event classes: Section 1.2.12 [break-event], page 42, Section 1.2.33 [line-break-event], page 45, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.83 [Page_turn_engraver], page 338, and Section 2.2.84 [Paper_column_engraver], page 338.

Properties:

`name` (symbol):
`'LineBreakEvent`
Name of this music object.

`types` (list):
`'(line-break-event break-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.38 LyricCombineMusic

Align lyrics to the start of notes.

Syntax: `\lyricsto` *voicename lyrics*

Properties:

`iterator-ctor` (procedure):
`ly:lyric-combine-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length` (moment):
`#<Mom 0>`
The duration of this music.

`name` (symbol):
`'LyricCombineMusic`
Name of this music object.

`types` (list):
`'(lyric-combine-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.39 LyricEvent

A lyric syllable. Must be entered in lyrics mode, i.e., `\lyrics { twinkle4 twinkle4 }` .

Event classes: Section 1.2.34 [lyric-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.64 [Lyric_engraver], page 331, and Section 2.2.65 [Lyric_performer], page 331.

Properties:

`iterator-ctor` (procedure):
`ly:rhythmic-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'LyricEvent`
Name of this music object.

`types` (list):
`'(rhythmic-event lyric-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.40 MarkEvent

Insert a rehearsal mark.

Syntax: `\mark` *marker*

Example: `\mark "A"`

Event classes: Section 1.2.35 [mark-event], page 45, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.66 [Mark_engraver], page 332.

Properties:

`name` (symbol):
`'MarkEvent`
Name of this music object.

`types` (list):
`'(mark-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.41 MeasureCounterEvent

Used to signal the start and end of a measure count.

Event classes: Section 1.2.36 [measure-counter-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.67 [Measure_counter_engraver], page 332.

Properties:

`name` (symbol):
`'MeasureCounterEvent`
Name of this music object.

`types` (list):
`'(measure-counter-event span-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.42 MultiMeasureRestEvent

Used internally by `MultiMeasureRestMusic` to signal rests.

Event classes: Section 1.2.38 [multi-measure-rest-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

Properties:

`iterator-ctor` (procedure):
`ly:rhythmic-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'MultiMeasureRestEvent`
Name of this music object.

`types` (list):
`'(event rhythmic-event multi-measure-rest-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.43 MultiMeasureRestMusic

Rests that may be compressed into Multi rests.

Syntax: `R2.*4` for 4 measures in 3/4 time.

Properties:

`elements-callback` (procedure):
`mm-rest-child-list`
Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:sequential-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'MultiMeasureRestMusic`
Name of this music object.

`types` (list):
`'(multi-measure-rest)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.44 MultiMeasureTextEvent

Texts on multi measure rests.

Syntax: `R-\markup { \roman "bla" }`

Note the explicit font switch.

Event classes: Section 1.2.39 [multi-measure-text-event], page 45, Section 1.2.40 [music-event], page 45, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

Properties:

`name` (symbol):
`'MultiMeasureTextEvent`
Name of this music object.

`types` (list):
`'(post-event event multi-measure-text-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.45 Music

Generic type for music expressions.

Properties:

`name` (symbol):
`'Music`
Name of this music object.

`types` (list):
`'()`
The types of this music object; determines by what engraver this music expression is processed.

1.1.46 NoteEvent

A note.

Outside of chords, any events in `articulations` with a listener are broadcast like chord articulations, the others are retained.

For iteration inside of chords, See Section 1.1.25 [EventChord], page 11.

Event classes: Section 1.2.37 [melodic-event], page 45, Section 1.2.40 [music-event], page 45, Section 1.2.41 [note-event], page 46, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.15 [Chord_name_engraver], page 313, Section 2.2.20 [Completion_heads_engraver], page 315, Section 2.2.30 [Drum_note_performer], page 319, Section 2.2.31 [Drum_notes_engraver], page 320, Section 2.2.44 [Fretboard_engraver], page 324, Section 2.2.77 [Note_heads_engraver], page 336, Section 2.2.78 [Note_name_engraver], page 336, Section 2.2.79 [Note_performer], page 337, Section 2.2.86 [Part_combine_engraver], page 339, Section 2.2.88 [Phrasing_slur_engraver], page 340, Section 2.2.105 [Slur_engraver], page 345, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

Properties:

`iterator-ctor` (procedure):
`ly:rhythmic-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'NoteEvent`
Name of this music object.

`types` (list):
`'(event note-event rhythmic-event melodic-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.47 NoteGroupingEvent

Start or stop grouping brackets.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.42 [note-grouping-event], page 46, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.53 [Horizontal_bracket_engraver], page 328.

Properties:

`name` (symbol):
`'NoteGroupingEvent`
Name of this music object.

`types` (list):
`'(post-event event note-grouping-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.48 OttavaMusic

Start or stop an ottava bracket.

Properties:

`elements-callback` (procedure):
`make-ottava-set`
Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:sequential-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'OttavaMusic`
Name of this music object.

`types` (list):
`'(ottava-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.49 OverrideProperty

Extend the definition of a graphical object.

Syntax: `\override` [*context* `.`] *object property* `=` *value*

Properties:

`iterator-ctor` (procedure):
`ly:push-property-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'OverrideProperty`

Name of this music object.

`types` (list):
```
'(layout-instruction-event
   override-property-event)
```

The types of this music object; determines by what engraver this music expression is processed.

`untransposable` (boolean):
`#t`

If set, this music is not transposed.

1.1.50 PageBreakEvent

Allow, forbid or force a page break.

Event classes: Section 1.2.12 [break-event], page 42, Section 1.2.40 [music-event], page 45, Section 1.2.43 [page-break-event], page 46, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.83 [Page_turn_engraver], page 338, and Section 2.2.84 [Paper_column_engraver], page 338.

Properties:

`name` (symbol):
`'PageBreakEvent`

Name of this music object.

`types` (list):
`'(break-event page-break-event event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.51 PageTurnEvent

Allow, forbid or force a page turn.

Event classes: Section 1.2.12 [break-event], page 42, Section 1.2.40 [music-event], page 45, Section 1.2.44 [page-turn-event], page 46, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.83 [Page_turn_engraver], page 338, and Section 2.2.84 [Paper_column_engraver], page 338.

Properties:

`name` (symbol):
`'PageTurnEvent`

Name of this music object.

`types` (list):
`'(break-event page-turn-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.52 PartCombineMusic

Combine two parts on a staff, either merged or as separate voices.

Properties:

`iterator-ctor` (procedure):
`ly:part-combine-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-sequence::maximum-length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'PartCombineMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-sequence::minimum-start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(part-combine-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.53 PartCombinePartMusic

A part to be combined with other parts on a staff.

Properties:

`iterator-ctor` (procedure):
`ly:part-combine-part-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'PartCombinePartMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(part-combine-part-music music-wrapper-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.54 PartialSet

Create an anacrusis or upbeat (partial measure).

Properties:

`iterator-ctor` (procedure):
`ly:partial-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-sequence::cumulative-length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'PartialSet`

Name of this music object.

`types` (list):
`'(partial-set)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.55 PercentEvent

Used internally to signal percent repeats.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.47 [percent-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.87 [Percent_repeat_engraver], page 339.

Properties:

`name` (symbol):
`'PercentEvent`

Name of this music object.

`types` (list):
`'(event percent-event rhythmic-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.56 PercentRepeatedMusic

Repeats encoded by percents and slashes.

Properties:

`iterator-ctor` (procedure):
`ly:percent-repeat-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:repeated-music::unfolded-music-length`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'PercentRepeatedMusic`
Name of this music object.

`start-callback` (procedure):
`ly:repeated-music::first-start`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(repeated-music percent-repeated-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.57 PesOrFlexaEvent

Within a ligature, mark the previous and the following note to form a pes (if melody goes up) or a flexa (if melody goes down).

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.48 [pes-or-flexa-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.134 [Vaticana_ligature_engraver], page 354.

Properties:

`name` (symbol):
`'PesOrFlexaEvent`
Name of this music object.

`types` (list):
`'(pes-or-flexa-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.58 PhrasingSlurEvent

Start or end phrasing slur.

Syntax: *note*`\(` and *note*`\)`

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.49 [phrasing-slur-event], page 47, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.88 [Phrasing_slur_engraver], page 340.

Properties:

`name` (symbol):
`'PhrasingSlurEvent`
Name of this music object.

`types` (list):
`'(post-event span-event event phrasing-slur-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.59 PostEvents

Container for several postevents.

This can be used to package several events into a single one. Should not be seen outside of the parser.

Properties:

`name` (symbol):
`'PostEvents`
Name of this music object.

`types` (list):
`'(post-event post-event-wrapper)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.60 PropertySet

Set a context property.

Syntax: `\set` *`context.prop`* `=` *`scheme-val`*

Properties:

`iterator-ctor` (procedure):
`ly:property-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'PropertySet`
Name of this music object.

`types` (list):
`'(layout-instruction-event)`
The types of this music object; determines by what engraver this music expression is processed.

`untransposable` (boolean):
`#t`
If set, this music is not transposed.

1.1.61 PropertyUnset

Restore the default setting for a context property. See Section 1.1.60 [PropertySet], page 24.

Syntax: `\unset` *`context.prop`*

Properties:

`iterator-ctor` (procedure):
`ly:property-unset-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'PropertyUnset`
Name of this music object.

`types` (list):
`'(layout-instruction-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.62 QuoteMusic

Quote preprocessed snippets of music.

Properties:

`iterator-ctor` (procedure):
`ly:music-wrapper-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'QuoteMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(music-wrapper-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.63 RelativeOctaveCheck

Check if a pitch is in the correct octave.

Properties:

`name` (symbol):
`'RelativeOctaveCheck`
Name of this music object.

`to-relative-callback` (procedure):
`ly:relative-octave-check::relative-callback`
How to transform a piece of music to relative pitches.

`types` (list):
`'(relative-octave-check)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.64 RelativeOctaveMusic

Music in which the assignment of octaves is complete.

Properties:

`iterator-ctor` (procedure):
`ly:music-wrapper-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
: `ly:music-wrapper::length-callback`
: How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
: `'RelativeOctaveMusic`
: Name of this music object.

`start-callback` (procedure):
: `ly:music-wrapper::start-callback`
: Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`to-relative-callback` (procedure):
: `ly:relative-octave-music::relative-callback`
: How to transform a piece of music to relative pitches.

`types` (list):
: `'(music-wrapper-music relative-octave-music)`
: The types of this music object; determines by what engraver this music expression is processed.

1.1.65 RepeatSlashEvent

Used internally to signal beat repeats.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.50 [repeat-slash-event], page 47, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.104 [Slash_repeat_engraver], page 345.

Properties:

`name` (symbol):
: `'RepeatSlashEvent`
: Name of this music object.

`types` (list):
: `'(event repeat-slash-event rhythmic-event)`
: The types of this music object; determines by what engraver this music expression is processed.

1.1.66 RepeatTieEvent

Ties for starting a second volta bracket.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.51 [repeat-tie-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.96 [Repeat_tie_engraver], page 342.

Properties:

`name` (symbol):
: `'RepeatTieEvent`
: Name of this music object.

`types` (list):
: `'(post-event event repeat-tie-event)`
: The types of this music object; determines by what engraver this music expression is processed.

1.1.67 RepeatedMusic

Repeat music in different ways.

Properties:

`name` (symbol):
`'RepeatedMusic`
Name of this music object.

`types` (list):
`'(repeated-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.68 RestEvent

A Rest.

Syntax: `r4` for a quarter rest.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.52 [rest-event], page 47, Section 1.2.53 [rhythmic-event], page 47, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.15 [Chord_name_engraver], page 313, Section 2.2.21 [Completion_rest_engraver], page 316, Section 2.2.37 [Figured_bass_engraver], page 322, and Section 2.2.98 [Rest_engraver], page 343.

Properties:

`iterator-ctor` (procedure):
`ly:rhythmic-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'RestEvent`
Name of this music object.

`types` (list):
`'(event rhythmic-event rest-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.69 RevertProperty

The opposite of Section 1.1.49 [OverrideProperty], page 20: remove a previously added property from a graphical object definition.

Properties:

`iterator-ctor` (procedure):
`ly:pop-property-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'RevertProperty`
Name of this music object.

`types` (list):
`'(layout-instruction-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.70 ScriptEvent

Add an articulation mark to a note.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.54 [script-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Not accepted by any engraver or performer.

Properties:

`name` (symbol):
`'ScriptEvent`

Name of this music object.

`types` (list):
`'(event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.71 SequentialMusic

Music expressions concatenated.

Syntax: `\sequential { ... }` or simply `{ ... }`

Properties:

`elements-callback` (procedure):
`#<procedure #f (m)>`

Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:sequential-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-sequence::cumulative-length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'SequentialMusic`

Name of this music object.

`start-callback` (procedure):
`ly:music-sequence::first-start-callback`

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(sequential-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.72 SimultaneousMusic

Music playing together.

Syntax: `\simultaneous { ... }` or `<< ... >>`

Properties:

`iterator-ctor` (procedure):
`ly:simultaneous-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-sequence::maximum-length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'SimultaneousMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-sequence::minimum-start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`to-relative-callback` (procedure):
`ly:music-sequence::simultaneous-relative-callback`
How to transform a piece of music to relative pitches.

`types` (list):
`'(simultaneous-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.73 SkipEvent

Filler that takes up duration, but does not print anything.

Syntax: `s4` for a skip equivalent to a quarter rest.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.53 [rhythmic-event], page 47, Section 1.2.55 [skip-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Not accepted by any engraver or performer.

Properties:

`iterator-ctor` (procedure):
`ly:rhythmic-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'SkipEvent`
Name of this music object.

`types` (list):
`'(event rhythmic-event skip-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.74 SkipMusic

Filler that takes up duration, does not print anything, and also does not create staves or voices implicitly.

Syntax: `\skip` *duration*

Properties:

`iterator-ctor` (procedure):
`ly:simple-music-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-duration-length`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'SkipMusic`
Name of this music object.

`types` (list):
`'(event skip-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.75 SlurEvent

Start or end slur.

Syntax: *note*`(` and *note*`)`

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.56 [slur-event], page 48, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.105 [Slur_engraver], page 345, and Section 2.2.106 [Slur_performer], page 345.

Properties:

`name` (symbol):
`'SlurEvent`
Name of this music object.

`types` (list):
`'(post-event span-event event slur-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.76 SoloOneEvent

Print ‘Solo 1’.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.45 [part-combine-event], page 47, Section 1.2.57 [solo-one-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.86 [Part_combine_engraver], page 339.

Properties:

`name` (symbol):
`'SoloOneEvent`
Name of this music object.

`part-combine-status` (symbol):
`'solo1`
Change to what kind of state? Options are `solo1`, `solo2` and `unisono`.

`types` (list):
`'(event part-combine-event solo-one-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.77 SoloTwoEvent

Print 'Solo 2'.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.45 [part-combine-event], page 47, Section 1.2.58 [solo-two-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.86 [Part_combine_engraver], page 339.

Properties:

`name` (symbol):
`'SoloTwoEvent`
Name of this music object.

`part-combine-status` (symbol):
`'solo2`
Change to what kind of state? Options are `solo1`, `solo2` and `unisono`.

`types` (list):
`'(event part-combine-event solo-two-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.78 SostenutoEvent

Depress or release sostenuto pedal.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.46 [pedal-event], page 47, Section 1.2.59 [sostenuto-event], page 48, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

Properties:

`name` (symbol):
`'SostenutoEvent`
Name of this music object.

`types` (list):
`'(post-event event pedal-event sostenuto-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.79 SpacingSectionEvent

Start a new spacing section.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.60 [spacing-section-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.107 [Spacing_engraver], page 346.

Properties:

`name` (symbol):
`'SpacingSectionEvent`
Name of this music object.

`types` (list):
`'(event spacing-section-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.80 SpanEvent

Event for anything that is started at a different time than stopped.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, and Section 1.2.64 [StreamEvent], page 49.

Not accepted by any engraver or performer.

Properties:

`name` (symbol):
`'SpanEvent`
Name of this music object.

`types` (list):
`'(event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.81 StaffSpanEvent

Start or stop a staff symbol.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, Section 1.2.63 [staff-span-event], page 49, and Section 1.2.64 [StreamEvent], page 49.

Accepted by: Section 2.2.115 [Staff_symbol_engraver], page 347.

Properties:

`name` (symbol):
`'StaffSpanEvent`
Name of this music object.

`types` (list):
`'(event span-event staff-span-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.82 StringNumberEvent

Specify on which string to play this note.

Syntax: `\`*`number`*

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.65 [string-number-event], page 49.

Accepted by: Section 2.2.44 [Fretboard_engraver], page 324, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

Properties:

`name` (symbol):
`'StringNumberEvent`
Name of this music object.

`types` (list):
`'(post-event string-number-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.83 StrokeFingerEvent

Specify with which finger to pluck a string.

Syntax: `\rightHandFinger` *text*

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.66 [stroke-finger-event], page 50.

Not accepted by any engraver or performer.

Properties:

`name` (symbol):
`'StrokeFingerEvent`
Name of this music object.

`types` (list):
`'(post-event stroke-finger-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.84 SustainEvent

Depress or release sustain pedal.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.46 [pedal-event], page 47, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.67 [sustain-event], page 50.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

Properties:

`name` (symbol):
`'SustainEvent`
Name of this music object.

`types` (list):
`'(post-event event pedal-event sustain-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.85 TempoChangeEvent

A metronome mark or tempo indication.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.68 [tempo-change-event], page 50.

Accepted by: Section 2.2.72 [Metronome_mark_engraver], page 333.

Properties:

`name` (symbol):
`'TempoChangeEvent`
Name of this music object.

`types` (list):
`'(event tempo-change-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.86 TextScriptEvent

Print text.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.54 [script-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.69 [text-script-event], page 50.

Accepted by: Section 2.2.124 [Text_engraver], page 350.

Properties:

`name` (symbol):
`'TextScriptEvent`
Name of this music object.

`types` (list):
`'(post-event script-event text-script-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.87 TextSpanEvent

Start a text spanner, for example, an octavation.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.70 [text-span-event], page 50.

Accepted by: Section 2.2.125 [Text_spanner_engraver], page 351.

Properties:

`name` (symbol):
`'TextSpanEvent`
Name of this music object.

`types` (list):
`'(post-event span-event event text-span-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.88 TieEvent

A tie.

Syntax: *note*`-~`

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.71 [tie-event], page 50.

Accepted by: Section 2.2.79 [Note_performer], page 337, Section 2.2.126 [Tie_engraver], page 351, and Section 2.2.127 [Tie_performer], page 351.

Properties:

`name` (symbol):
`'TieEvent`
Name of this music object.

`types` (list):
`'(post-event tie-event event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.89 TimeScaledMusic

Multiply durations, as in tuplets.

Syntax: `\times` *`fraction music`*, e.g., `\times 2/3 { ... }` for triplets.

Properties:

`iterator-ctor` (procedure):
`ly:tuplet-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'TimeScaledMusic`
Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(time-scaled-music music-wrapper-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.90 TimeSignatureEvent

An event created when setting a new time signature

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.72 [time-signature-event], page 50.

Accepted by: Section 2.2.128 [Time_signature_engraver], page 352.

Properties:

`name` (symbol):
`'TimeSignatureEvent`
Name of this music object.

`types` (list):
`'(event time-signature-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.91 TimeSignatureMusic

Set a new time signature

Properties:

`elements-callback` (procedure):
`make-time-signature-set`

Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:sequential-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`name` (symbol):
`'TimeSignatureMusic`

Name of this music object.

`types` (list):
`'(time-signature-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.92 TransposedMusic

Music that has been transposed.

Properties:

`iterator-ctor` (procedure):
`ly:music-wrapper-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'TransposedMusic`

Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`to-relative-callback` (procedure):
`ly:relative-octave-music::no-relative-callback`

How to transform a piece of music to relative pitches.

`types` (list):
`'(music-wrapper-music transposed-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.93 TremoloEvent

Unmeasured tremolo.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.73 [tremolo-event], page 50.

Accepted by: Section 2.2.118 [Stem_engraver], page 348.

Properties:

`name` (symbol):
`'TremoloEvent`
Name of this music object.

`types` (list):
`'(post-event event tremolo-event)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.94 TremoloRepeatedMusic

Repeated notes denoted by tremolo beams.

Properties:

`iterator-ctor` (procedure):
`ly:chord-tremolo-iterator::constructor`
Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:repeated-music::unfolded-music-length`
How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'TremoloRepeatedMusic`
Name of this music object.

`start-callback` (procedure):
`ly:repeated-music::first-start`
Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(repeated-music tremolo-repeated-music)`
The types of this music object; determines by what engraver this music expression is processed.

1.1.95 TremoloSpanEvent

Tremolo over two stems.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.74 [tremolo-span-event], page 50.

Accepted by: Section 2.2.16 [Chord_tremolo_engraver], page 314.

Properties:

`name` (symbol):
`'TremoloSpanEvent`
Name of this music object.

`types` (list):
`'(event span-event tremolo-span-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.96 TrillSpanEvent

Start a trill spanner.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.75 [trill-span-event], page 50.

Accepted by: Section 2.2.131 [Trill_spanner_engraver], page 353.

Properties:

`name` (symbol):
`'TrillSpanEvent`

Name of this music object.

`types` (list):
`'(post-event span-event event trill-span-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.97 TupletSpanEvent

Used internally to signal where tuplet brackets start and stop.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.76 [tuplet-span-event], page 51.

Accepted by: Section 2.2.118 [Stem_engraver], page 348, and Section 2.2.132 [Tuplet_engraver], page 353.

Properties:

`name` (symbol):
`'TupletSpanEvent`

Name of this music object.

`types` (list):
`'(tuplet-span-event span-event event post-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.98 UnaCordaEvent

Depress or release una-corda pedal.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.46 [pedal-event], page 47, Section 1.2.62 [span-event], page 48, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.77 [una-corda-event], page 51.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

Properties:

`name` (symbol):
`'UnaCordaEvent`

Name of this music object.

`types` (list):
`'(post-event event pedal-event una-corda-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.99 UnfoldedRepeatedMusic

Repeated music which is fully written (and played) out.

Properties:

`elements-callback` (procedure):
`make-unfolded-set`

Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:sequential-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:repeated-music::unfolded-music-length`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'UnfoldedRepeatedMusic`

Name of this music object.

`start-callback` (procedure):
`ly:repeated-music::first-start`

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(repeated-music unfolded-repeated-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.100 UnisonoEvent

Print ‘a 2’.

Event classes: Section 1.2.40 [music-event], page 45, Section 1.2.45 [part-combine-event], page 47, Section 1.2.64 [StreamEvent], page 49, and Section 1.2.78 [unisono-event], page 51.

Accepted by: Section 2.2.86 [Part_combine_engraver], page 339.

Properties:

`name` (symbol):
`'UnisonoEvent`

Name of this music object.

`part-combine-status` (symbol):
`'unisono`

Change to what kind of state? Options are `solo1`, `solo2` and `unisono`.

`types` (list):
`'(event part-combine-event unisono-event)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.101 UnrelativableMusic

Music that cannot be converted from relative to absolute notation. For example, transposed music.

Properties:

`iterator-ctor` (procedure):
`ly:music-wrapper-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:music-wrapper::length-callback`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'UnrelativableMusic`

Name of this music object.

`start-callback` (procedure):
`ly:music-wrapper::start-callback`

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`to-relative-callback` (procedure):
`ly:relative-octave-music::no-relative-callback`

How to transform a piece of music to relative pitches.

`types` (list):
`'(music-wrapper-music unrelativable-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.102 VoiceSeparator

Separate polyphonic voices in simultaneous music.

Syntax: `\\`

Properties:

`name` (symbol):
`'VoiceSeparator`

Name of this music object.

`types` (list):
`'(separator)`

The types of this music object; determines by what engraver this music expression is processed.

1.1.103 VoltaRepeatedMusic

Repeats with alternatives placed sequentially.

Properties:

`elements-callback` (procedure):
`make-volta-set`

Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`iterator-ctor` (procedure):
`ly:volta-repeat-iterator::constructor`

Function to construct a `music-event-iterator` object for this music.

`length-callback` (procedure):
`ly:repeated-music::volta-music-length`

How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`name` (symbol):
`'VoltaRepeatedMusic`

Name of this music object.

`start-callback` (procedure):
`ly:repeated-music::first-start`

Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`types` (list):
`'(repeated-music volta-repeated-music)`

The types of this music object; determines by what engraver this music expression is processed.

1.2 Music classes

1.2.1 absolute-dynamic-event

Music event type `absolute-dynamic-event` is in music objects of type Section 1.1.1 [AbsoluteDynamicEvent], page 2.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320, and Section 2.2.34 [Dynamic_performer], page 321.

1.2.2 alternative-event

Music event type `alternative-event` is in music objects of type Section 1.1.2 [AlternativeEvent], page 2.

Accepted by: Section 2.2.8 [Bar_number_engraver], page 310.

1.2.3 annotate-output-event

Music event type `annotate-output-event` is in music objects of type Section 1.1.3 [AnnotateOutputEvent], page 2.

Accepted by: Section 2.2.6 [Balloon_engraver], page 310.

1.2.4 apply-output-event

Music event type `apply-output-event` is in music objects of type Section 1.1.5 [ApplyOutputEvent], page 3.

Accepted by: Section 2.2.82 [Output_property_engraver], page 338.

1.2.5 arpeggio-event

Music event type `arpeggio-event` is in music objects of type Section 1.1.6 [ArpeggioEvent], page 3.

Accepted by: Section 2.2.3 [Arpeggio_engraver], page 308.

1.2.6 articulation-event

Music event type `articulation-event` is in music objects of type Section 1.1.7 [ArticulationEvent], page 4.

Accepted by: Section 2.2.79 [Note_performer], page 337, and Section 2.2.101 [Script_engraver], page 344.

1.2.7 bass-figure-event

Music event type `bass-figure-event` is in music objects of type Section 1.1.10 [BassFigureEvent], page 5.

Accepted by: Section 2.2.37 [Figured_bass_engraver], page 322.

1.2.8 beam-event

Music event type `beam-event` is in music objects of type Section 1.1.11 [BeamEvent], page 5.

Accepted by: Section 2.2.10 [Beam_engraver], page 312, Section 2.2.11 [Beam_performer], page 312, and Section 2.2.47 [Grace_beam_engraver], page 326.

1.2.9 beam-forbid-event

Music event type `beam-forbid-event` is in music objects of type Section 1.1.12 [BeamForbidEvent], page 6.

Accepted by: Section 2.2.4 [Auto_beam_engraver], page 308, and Section 2.2.46 [Grace_auto_beam_engraver], page 325.

1.2.10 bend-after-event

Music event type `bend-after-event` is in music objects of type Section 1.1.13 [BendAfterEvent], page 6.

Accepted by: Section 2.2.12 [Bend_engraver], page 312.

1.2.11 break-dynamic-span-event

Music event type `break-dynamic-span-event` is in music objects of type Section 1.1.14 [BreakDynamicSpanEvent], page 6.

Not accepted by any engraver or performer.

1.2.12 break-event

Music event type `break-event` is in music objects of type Section 1.1.37 [LineBreakEvent], page 15, Section 1.1.50 [PageBreakEvent], page 20, and Section 1.1.51 [PageTurnEvent], page 20.

Accepted by: Section 2.2.83 [Page_turn_engraver], page 338, and Section 2.2.84 [Paper_column_engraver], page 338.

1.2.13 break-span-event

Music event type `break-span-event` is in music objects of type Section 1.1.14 [BreakDynamicSpanEvent], page 6.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320.

1.2.14 breathing-event

Music event type `breathing-event` is in music objects of type Section 1.1.15 [BreathingEvent], page 7.

Accepted by: Section 2.2.14 [Breathing_sign_engraver], page 313, and Section 2.2.79 [Note_performer], page 337.

1.2.15 cluster-note-event

Music event type `cluster-note-event` is in music objects of type Section 1.1.16 [ClusterNoteEvent], page 7.

Accepted by: Section 2.2.18 [Cluster_spanner_engraver], page 315.

1.2.16 completize-extender-event

Music event type `completize-extender-event` is in music objects of type Section 1.1.17 [CompletizeExtenderEvent], page 8.

Accepted by: Section 2.2.36 [Extender_engraver], page 322.

1.2.17 crescendo-event

Music event type `crescendo-event` is in music objects of type Section 1.1.20 [CrescendoEvent], page 9.

Accepted by: Section 2.2.34 [Dynamic_performer], page 321.

1.2.18 decrescendo-event

Music event type `decrescendo-event` is in music objects of type Section 1.1.21 [DecrescendoEvent], page 9.

Accepted by: Section 2.2.34 [Dynamic_performer], page 321.

1.2.19 double-percent-event

Music event type `double-percent-event` is in music objects of type Section 1.1.22 [DoublePercentEvent], page 10.

Accepted by: Section 2.2.29 [Double_percent_repeat_engraver], page 319.

1.2.20 dynamic-event

Music event type `dynamic-event` is in music objects of type Section 1.1.1 [AbsoluteDynamicEvent], page 2.

Not accepted by any engraver or performer.

1.2.21 episema-event

Music event type `episema-event` is in music objects of type Section 1.1.23 [EpisemaEvent], page 10.

Accepted by: Section 2.2.35 [Episema_engraver], page 321.

1.2.22 extender-event

Music event type `extender-event` is in music objects of type Section 1.1.26 [ExtenderEvent], page 11.

Accepted by: Section 2.2.36 [Extender_engraver], page 322.

1.2.23 fingering-event

Music event type `fingering-event` is in music objects of type Section 1.1.27 [FingeringEvent], page 12.

Accepted by: Section 2.2.40 [Fingering_engraver], page 323, Section 2.2.44 [Fretboard_engraver], page 324, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

1.2.24 footnote-event

Music event type `footnote-event` is in music objects of type Section 1.1.28 [FootnoteEvent], page 12.

Not accepted by any engraver or performer.

1.2.25 glissando-event

Music event type `glissando-event` is in music objects of type Section 1.1.29 [GlissandoEvent], page 12.

Accepted by: Section 2.2.45 [Glissando_engraver], page 325.

1.2.26 harmonic-event

Music event type `harmonic-event` is in music objects of type Section 1.1.31 [HarmonicEvent], page 13.

Not accepted by any engraver or performer.

1.2.27 hyphen-event

Music event type `hyphen-event` is in music objects of type Section 1.1.32 [HyphenEvent], page 13.

Accepted by: Section 2.2.54 [Hyphen_engraver], page 328.

1.2.28 key-change-event

Music event type `key-change-event` is in music objects of type Section 1.1.33 [KeyChangeEvent], page 14.

Accepted by: Section 2.2.58 [Key_engraver], page 329, and Section 2.2.59 [Key_performer], page 330.

1.2.29 label-event

Music event type `label-event` is in music objects of type Section 1.1.34 [LabelEvent], page 14.

Accepted by: Section 2.2.84 [Paper_column_engraver], page 338.

1.2.30 laissez-vibrer-event

Music event type `laissez-vibrer-event` is in music objects of type Section 1.1.35 [LaissezVibrerEvent], page 14.

Accepted by: Section 2.2.61 [Laissez_vibrer_engraver], page 330.

1.2.31 layout-instruction-event

Music event type `layout-instruction-event` is in music objects of type Section 1.1.5 [ApplyOutputEvent], page 3.

Not accepted by any engraver or performer.

1.2.32 ligature-event

Music event type `ligature-event` is in music objects of type Section 1.1.36 [LigatureEvent], page 15.

Accepted by: Section 2.2.60 [Kievan_ligature_engraver], page 330, Section 2.2.63 [Ligature_bracket_engraver], page 331, Section 2.2.70 [Mensural_ligature_engraver], page 333, and Section 2.2.134 [Vaticana_ligature_engraver], page 354.

1.2.33 line-break-event

Music event type `line-break-event` is in music objects of type Section 1.1.37 [LineBreakEvent], page 15.

Not accepted by any engraver or performer.

1.2.34 lyric-event

Music event type `lyric-event` is in music objects of type Section 1.1.39 [LyricEvent], page 16.

Accepted by: Section 2.2.64 [Lyric_engraver], page 331, and Section 2.2.65 [Lyric_performer], page 331.

1.2.35 mark-event

Music event type `mark-event` is in music objects of type Section 1.1.40 [MarkEvent], page 16.

Accepted by: Section 2.2.66 [Mark_engraver], page 332.

1.2.36 measure-counter-event

Music event type `measure-counter-event` is in music objects of type Section 1.1.41 [MeasureCounterEvent], page 17.

Accepted by: Section 2.2.67 [Measure_counter_engraver], page 332.

1.2.37 melodic-event

Music event type `melodic-event` is in music objects of type Section 1.1.16 [ClusterNoteEvent], page 7, and Section 1.1.46 [NoteEvent], page 18.

Not accepted by any engraver or performer.

1.2.38 multi-measure-rest-event

Music event type `multi-measure-rest-event` is in music objects of type Section 1.1.42 [MultiMeasureRestEvent], page 17.

Accepted by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

1.2.39 multi-measure-text-event

Music event type `multi-measure-text-event` is in music objects of type Section 1.1.44 [MultiMeasureTextEvent], page 18.

Accepted by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

1.2.40 music-event

Music event type `music-event` is in music objects of type Section 1.1.1 [AbsoluteDynamicEvent], page 2, Section 1.1.2 [AlternativeEvent], page 2, Section 1.1.3 [AnnotateOutputEvent], page 2, Section 1.1.5 [ApplyOutputEvent], page 3, Section 1.1.6 [ArpeggioEvent], page 3, Section 1.1.7 [ArticulationEvent], page 4, Section 1.1.10 [BassFigureEvent], page 5, Section 1.1.11 [BeamEvent], page 5, Section 1.1.12 [BeamForbidEvent], page 6, Section 1.1.13 [BendAfterEvent], page 6, Section 1.1.14 [BreakDynamicSpanEvent], page 6, Section 1.1.15 [BreathingEvent], page 7, Section 1.1.16 [ClusterNoteEvent], page 7, Section 1.1.17

[CompletizeExtenderEvent], page 8, Section 1.1.20 [CrescendoEvent], page 9, Section 1.1.21 [DecrescendoEvent], page 9, Section 1.1.22 [DoublePercentEvent], page 10, Section 1.1.23 [EpisemaEvent], page 10, Section 1.1.26 [ExtenderEvent], page 11, Section 1.1.27 [FingeringEvent], page 12, Section 1.1.28 [FootnoteEvent], page 12, Section 1.1.29 [GlissandoEvent], page 12, Section 1.1.31 [HarmonicEvent], page 13, Section 1.1.32 [HyphenEvent], page 13, Section 1.1.33 [KeyChangeEvent], page 14, Section 1.1.34 [LabelEvent], page 14, Section 1.1.35 [LaissezVibrerEvent], page 14, Section 1.1.36 [LigatureEvent], page 15, Section 1.1.37 [LineBreakEvent], page 15, Section 1.1.39 [LyricEvent], page 16, Section 1.1.40 [MarkEvent], page 16, Section 1.1.41 [MeasureCounterEvent], page 17, Section 1.1.42 [MultiMeasureRestEvent], page 17, Section 1.1.44 [MultiMeasureTextEvent], page 18, Section 1.1.46 [NoteEvent], page 18, Section 1.1.47 [NoteGroupingEvent], page 19, Section 1.1.50 [PageBreakEvent], page 20, Section 1.1.51 [PageTurnEvent], page 20, Section 1.1.55 [PercentEvent], page 22, Section 1.1.57 [PesOrFlexaEvent], page 23, Section 1.1.58 [PhrasingSlurEvent], page 23, Section 1.1.65 [RepeatSlashEvent], page 26, Section 1.1.66 [RepeatTieEvent], page 26, Section 1.1.68 [RestEvent], page 27, Section 1.1.70 [ScriptEvent], page 28, Section 1.1.73 [SkipEvent], page 29, Section 1.1.75 [SlurEvent], page 30, Section 1.1.76 [SoloOneEvent], page 30, Section 1.1.77 [SoloTwoEvent], page 31, Section 1.1.78 [SostenutoEvent], page 31, Section 1.1.79 [SpacingSectionEvent], page 31, Section 1.1.80 [SpanEvent], page 32, Section 1.1.81 [StaffSpanEvent], page 32, Section 1.1.82 [StringNumberEvent], page 32, Section 1.1.83 [StrokeFingerEvent], page 33, Section 1.1.84 [SustainEvent], page 33, Section 1.1.85 [TempoChangeEvent], page 33, Section 1.1.86 [TextScriptEvent], page 34, Section 1.1.87 [TextSpanEvent], page 34, Section 1.1.88 [TieEvent], page 34, Section 1.1.90 [TimeSignatureEvent], page 35, Section 1.1.93 [TremoloEvent], page 37, Section 1.1.95 [TremoloSpanEvent], page 37, Section 1.1.96 [TrillSpanEvent], page 38, Section 1.1.97 [TupletSpanEvent], page 38, Section 1.1.98 [UnaCordaEvent], page 38, and Section 1.1.100 [UnisonoEvent], page 39.

Not accepted by any engraver or performer.

1.2.41 note-event

Music event type `note-event` is in music objects of type Section 1.1.46 [NoteEvent], page 18.

Accepted by: Section 2.2.15 [Chord_name_engraver], page 313, Section 2.2.20 [Completion_heads_engraver], page 315, Section 2.2.30 [Drum_note_performer], page 319, Section 2.2.31 [Drum_notes_engraver], page 320, Section 2.2.44 [Fretboard_engraver], page 324, Section 2.2.77 [Note_heads_engraver], page 336, Section 2.2.78 [Note_name_engraver], page 336, Section 2.2.79 [Note_performer], page 337, Section 2.2.86 [Part_combine_engraver], page 339, Section 2.2.88 [Phrasing_slur_engraver], page 340, Section 2.2.105 [Slur_engraver], page 345, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

1.2.42 note-grouping-event

Music event type `note-grouping-event` is in music objects of type Section 1.1.47 [NoteGroupingEvent], page 19.

Accepted by: Section 2.2.53 [Horizontal_bracket_engraver], page 328.

1.2.43 page-break-event

Music event type `page-break-event` is in music objects of type Section 1.1.50 [PageBreakEvent], page 20.

Not accepted by any engraver or performer.

1.2.44 page-turn-event

Music event type `page-turn-event` is in music objects of type Section 1.1.51 [PageTurnEvent], page 20.

Not accepted by any engraver or performer.

1.2.45 part-combine-event

Music event type `part-combine-event` is in music objects of type Section 1.1.76 [SoloOneEvent], page 30, Section 1.1.77 [SoloTwoEvent], page 31, and Section 1.1.100 [UnisonoEvent], page 39.

Accepted by: Section 2.2.86 [Part_combine_engraver], page 339.

1.2.46 pedal-event

Music event type `pedal-event` is in music objects of type Section 1.1.78 [SostenutoEvent], page 31, Section 1.1.84 [SustainEvent], page 33, and Section 1.1.98 [UnaCordaEvent], page 38.

Not accepted by any engraver or performer.

1.2.47 percent-event

Music event type `percent-event` is in music objects of type Section 1.1.55 [PercentEvent], page 22.

Accepted by: Section 2.2.87 [Percent_repeat_engraver], page 339.

1.2.48 pes-or-flexa-event

Music event type `pes-or-flexa-event` is in music objects of type Section 1.1.57 [PesOrFlexaEvent], page 23.

Accepted by: Section 2.2.134 [Vaticana_ligature_engraver], page 354.

1.2.49 phrasing-slur-event

Music event type `phrasing-slur-event` is in music objects of type Section 1.1.58 [PhrasingSlurEvent], page 23.

Accepted by: Section 2.2.88 [Phrasing_slur_engraver], page 340.

1.2.50 repeat-slash-event

Music event type `repeat-slash-event` is in music objects of type Section 1.1.65 [RepeatSlashEvent], page 26.

Accepted by: Section 2.2.104 [Slash_repeat_engraver], page 345.

1.2.51 repeat-tie-event

Music event type `repeat-tie-event` is in music objects of type Section 1.1.66 [RepeatTieEvent], page 26.

Accepted by: Section 2.2.96 [Repeat_tie_engraver], page 342.

1.2.52 rest-event

Music event type `rest-event` is in music objects of type Section 1.1.68 [RestEvent], page 27.

Accepted by: Section 2.2.15 [Chord_name_engraver], page 313, Section 2.2.21 [Completion_rest_engraver], page 316, Section 2.2.37 [Figured_bass_engraver], page 322, and Section 2.2.98 [Rest_engraver], page 343.

1.2.53 rhythmic-event

Music event type `rhythmic-event` is in music objects of type Section 1.1.10 [BassFigureEvent], page 5, Section 1.1.16 [ClusterNoteEvent], page 7, Section 1.1.22 [DoublePercentEvent], page 10, Section 1.1.39 [LyricEvent], page 16, Section 1.1.42 [MultiMeasureRestEvent], page 17, Section 1.1.46 [NoteEvent], page 18, Section 1.1.65 [RepeatSlashEvent], page 26, Section 1.1.68 [RestEvent], page 27, and Section 1.1.73 [SkipEvent], page 29.

Not accepted by any engraver or performer.

1.2.54 script-event

Music event type `script-event` is in music objects of type Section 1.1.7 [ArticulationEvent], page 4, Section 1.1.70 [ScriptEvent], page 28, and Section 1.1.86 [TextScriptEvent], page 34.

Not accepted by any engraver or performer.

1.2.55 skip-event

Music event type `skip-event` is in music objects of type Section 1.1.73 [SkipEvent], page 29.

Not accepted by any engraver or performer.

1.2.56 slur-event

Music event type `slur-event` is in music objects of type Section 1.1.75 [SlurEvent], page 30.

Accepted by: Section 2.2.105 [Slur_engraver], page 345, and Section 2.2.106 [Slur_performer], page 345.

1.2.57 solo-one-event

Music event type `solo-one-event` is in music objects of type Section 1.1.76 [SoloOneEvent], page 30.

Not accepted by any engraver or performer.

1.2.58 solo-two-event

Music event type `solo-two-event` is in music objects of type Section 1.1.77 [SoloTwoEvent], page 31.

Not accepted by any engraver or performer.

1.2.59 sostenuto-event

Music event type `sostenuto-event` is in music objects of type Section 1.1.78 [SostenutoEvent], page 31.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

1.2.60 spacing-section-event

Music event type `spacing-section-event` is in music objects of type Section 1.1.79 [SpacingSectionEvent], page 31.

Accepted by: Section 2.2.107 [Spacing_engraver], page 346.

1.2.61 span-dynamic-event

Music event type `span-dynamic-event` is in music objects of type Section 1.1.20 [CrescendoEvent], page 9, and Section 1.1.21 [DecrescendoEvent], page 9.

Accepted by: Section 2.2.33 [Dynamic_engraver], page 320.

1.2.62 span-event

Music event type `span-event` is in music objects of type Section 1.1.11 [BeamEvent], page 5, Section 1.1.20 [CrescendoEvent], page 9, Section 1.1.21 [DecrescendoEvent], page 9, Section 1.1.23 [EpisemaEvent], page 10, Section 1.1.36 [LigatureEvent], page 15, Section 1.1.41 [MeasureCounterEvent], page 17, Section 1.1.58 [PhrasingSlurEvent], page 23, Section 1.1.75 [SlurEvent], page 30, Section 1.1.78 [SostenutoEvent], page 31, Section 1.1.80 [SpanEvent], page 32, Section 1.1.81 [StaffSpanEvent], page 32, Section 1.1.84 [SustainEvent], page 33, Section 1.1.87 [TextSpanEvent], page 34, Section 1.1.95 [TremoloSpanEvent], page 37,

Section 1.1.96 [TrillSpanEvent], page 38, Section 1.1.97 [TupletSpanEvent], page 38, and Section 1.1.98 [UnaCordaEvent], page 38.

Not accepted by any engraver or performer.

1.2.63 staff-span-event

Music event type `staff-span-event` is in music objects of type Section 1.1.81 [StaffSpanEvent], page 32.

Accepted by: Section 2.2.115 [Staff_symbol_engraver], page 347.

1.2.64 StreamEvent

Music event type `StreamEvent` is in music objects of type Section 1.1.1 [AbsoluteDynamicEvent], page 2, Section 1.1.2 [AlternativeEvent], page 2, Section 1.1.3 [AnnotateOutputEvent], page 2, Section 1.1.5 [ApplyOutputEvent], page 3, Section 1.1.6 [ArpeggioEvent], page 3, Section 1.1.7 [ArticulationEvent], page 4, Section 1.1.10 [BassFigureEvent], page 5, Section 1.1.11 [BeamEvent], page 5, Section 1.1.12 [BeamForbidEvent], page 6, Section 1.1.13 [BendAfterEvent], page 6, Section 1.1.14 [BreakDynamicSpanEvent], page 6, Section 1.1.15 [BreathingEvent], page 7, Section 1.1.16 [ClusterNoteEvent], page 7, Section 1.1.17 [CompletizeExtenderEvent], page 8, Section 1.1.20 [CrescendoEvent], page 9, Section 1.1.21 [DecrescendoEvent], page 9, Section 1.1.22 [DoublePercentEvent], page 10, Section 1.1.23 [EpisemaEvent], page 10, Section 1.1.26 [ExtenderEvent], page 11, Section 1.1.27 [FingeringEvent], page 12, Section 1.1.28 [FootnoteEvent], page 12, Section 1.1.29 [GlissandoEvent], page 12, Section 1.1.31 [HarmonicEvent], page 13, Section 1.1.32 [HyphenEvent], page 13, Section 1.1.33 [KeyChangeEvent], page 14, Section 1.1.34 [LabelEvent], page 14, Section 1.1.35 [LaissezVibrerEvent], page 14, Section 1.1.36 [LigatureEvent], page 15, Section 1.1.37 [LineBreakEvent], page 15, Section 1.1.39 [LyricEvent], page 16, Section 1.1.40 [MarkEvent], page 16, Section 1.1.41 [MeasureCounterEvent], page 17, Section 1.1.42 [MultiMeasureRestEvent], page 17, Section 1.1.44 [MultiMeasureTextEvent], page 18, Section 1.1.46 [NoteEvent], page 18, Section 1.1.47 [NoteGroupingEvent], page 19, Section 1.1.50 [PageBreakEvent], page 20, Section 1.1.51 [PageTurnEvent], page 20, Section 1.1.55 [PercentEvent], page 22, Section 1.1.57 [PesOrFlexaEvent], page 23, Section 1.1.58 [PhrasingSlurEvent], page 23, Section 1.1.65 [RepeatSlashEvent], page 26, Section 1.1.66 [RepeatTieEvent], page 26, Section 1.1.68 [RestEvent], page 27, Section 1.1.70 [ScriptEvent], page 28, Section 1.1.73 [SkipEvent], page 29, Section 1.1.75 [SlurEvent], page 30, Section 1.1.76 [SoloOneEvent], page 30, Section 1.1.77 [SoloTwoEvent], page 31, Section 1.1.78 [SostenutoEvent], page 31, Section 1.1.79 [SpacingSectionEvent], page 31, Section 1.1.80 [SpanEvent], page 32, Section 1.1.81 [StaffSpanEvent], page 32, Section 1.1.82 [StringNumberEvent], page 32, Section 1.1.83 [StrokeFingerEvent], page 33, Section 1.1.84 [SustainEvent], page 33, Section 1.1.85 [TempoChangeEvent], page 33, Section 1.1.86 [TextScriptEvent], page 34, Section 1.1.87 [TextSpanEvent], page 34, Section 1.1.88 [TieEvent], page 34, Section 1.1.90 [TimeSignatureEvent], page 35, Section 1.1.93 [TremoloEvent], page 37, Section 1.1.95 [TremoloSpanEvent], page 37, Section 1.1.96 [TrillSpanEvent], page 38, Section 1.1.97 [TupletSpanEvent], page 38, Section 1.1.98 [UnaCordaEvent], page 38, and Section 1.1.100 [UnisonoEvent], page 39.

Not accepted by any engraver or performer.

1.2.65 string-number-event

Music event type `string-number-event` is in music objects of type Section 1.1.82 [StringNumberEvent], page 32.

Accepted by: Section 2.2.44 [Fretboard_engraver], page 324, and Section 2.2.120 [Tab_note_heads_engraver], page 349.

1.2.66 stroke-finger-event

Music event type `stroke-finger-event` is in music objects of type Section 1.1.83 [StrokeFingerEvent], page 33.

Not accepted by any engraver or performer.

1.2.67 sustain-event

Music event type `sustain-event` is in music objects of type Section 1.1.84 [SustainEvent], page 33.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

1.2.68 tempo-change-event

Music event type `tempo-change-event` is in music objects of type Section 1.1.85 [TempoChangeEvent], page 33.

Accepted by: Section 2.2.72 [Metronome_mark_engraver], page 333.

1.2.69 text-script-event

Music event type `text-script-event` is in music objects of type Section 1.1.86 [TextScriptEvent], page 34.

Accepted by: Section 2.2.124 [Text_engraver], page 350.

1.2.70 text-span-event

Music event type `text-span-event` is in music objects of type Section 1.1.87 [TextSpanEvent], page 34.

Accepted by: Section 2.2.125 [Text_spanner_engraver], page 351.

1.2.71 tie-event

Music event type `tie-event` is in music objects of type Section 1.1.88 [TieEvent], page 34.

Accepted by: Section 2.2.79 [Note_performer], page 337, Section 2.2.126 [Tie_engraver], page 351, and Section 2.2.127 [Tie_performer], page 351.

1.2.72 time-signature-event

Music event type `time-signature-event` is in music objects of type Section 1.1.90 [TimeSignatureEvent], page 35.

Accepted by: Section 2.2.128 [Time_signature_engraver], page 352.

1.2.73 tremolo-event

Music event type `tremolo-event` is in music objects of type Section 1.1.93 [TremoloEvent], page 37.

Accepted by: Section 2.2.118 [Stem_engraver], page 348.

1.2.74 tremolo-span-event

Music event type `tremolo-span-event` is in music objects of type Section 1.1.95 [TremoloSpanEvent], page 37.

Accepted by: Section 2.2.16 [Chord_tremolo_engraver], page 314.

1.2.75 trill-span-event

Music event type `trill-span-event` is in music objects of type Section 1.1.96 [TrillSpanEvent], page 38.

Accepted by: Section 2.2.131 [Trill_spanner_engraver], page 353.

1.2.76 tuplet-span-event

Music event type `tuplet-span-event` is in music objects of type Section 1.1.97 [TupletSpanEvent], page 38.

Accepted by: Section 2.2.118 [Stem_engraver], page 348, and Section 2.2.132 [Tuplet_engraver], page 353.

1.2.77 una-corda-event

Music event type `una-corda-event` is in music objects of type Section 1.1.98 [UnaCordaEvent], page 38.

Accepted by: Section 2.2.90 [Piano_pedal_engraver], page 340, and Section 2.2.91 [Piano_pedal_performer], page 341.

1.2.78 unisono-event

Music event type `unisono-event` is in music objects of type Section 1.1.100 [UnisonoEvent], page 39.

Not accepted by any engraver or performer.

1.3 Music properties

`absolute-octave` (integer)
: The absolute octave for an octave check note.

`alteration` (number)
: Alteration for figured bass.

`alternative-dir` (direction)
: Indicates if an AlternativeMusic is the First (-1), Middle (0), or Last (1) of group of alternate endings.

`alternative-increment` (integer)
: The number of times an alternative's lettering should be incremented.

`articulation-type` (string)
: Key for script definitions alist.

  TODO: Consider making type into symbol.

`articulations` (list of music objects)
: Articulation events specifically for this note.

`associated-context` (string)
: Name of the context associated with this `\lyricsto` section.

`associated-context-type` (symbol)
: Type of the context associated with this `\lyricsto` section.

`augmented` (boolean)
: This figure is for an augmented figured bass (with `+` sign).

`augmented-slash` (boolean)
: This figure is for an augmented figured bass (back-slashed number).

`automatically-numbered` (boolean)
: Should a footnote be automatically numbered?

`autosplit-end` (boolean)
: Duration of event was truncated by automatic splitting in `Completion_heads_engraver`.

`bass` (boolean)
: Set if this note is a bass note in a chord.

`beat-structure` (list)
: A beatStructure to be used in autobeaming.

`bracket-start` (boolean)
: Start a bracket here.

  TODO: Use SpanEvents?

`bracket-stop` (boolean)
: Stop a bracket here.

`break-penalty` (number)
: Penalty for line break hint.

`break-permission` (symbol)
: Whether to allow, forbid or force a line break.

`cautionary` (boolean)
: If set, this alteration needs a cautionary accidental.

`change-to-id` (string)
: Name of the context to change to.

`change-to-type` (symbol)
: Type of the context to change to.

`class` (symbol)
: The class name of an event class.

`context` (context)
: The context to which an event is sent.

`context-change-list` (list)
: Context changes for `\autochange` or `\partcombine`.

`context-id` (string)
: Name of context.

`context-type` (symbol)
: Type of context.

`create-new` (boolean)
: Create a fresh context.

`delta-step` (number)
: How much should a fall change pitch?

`denominator` (integer)
: Denominator in a time signature.

`descend-only` (boolean)
: If set, this `\context` only descends in the context tree.

`digit` (integer)
: Digit for fingering.

`diminished` (boolean)
: This bass figure should be slashed.

`direction` (direction)
: Print this up or down?

`drum-type` (symbol)
: Which percussion instrument to play this note on.

`duration` (duration)
: Duration of this note or lyric.

`element` (music)
: The single child of a Music_wrapper music object, or the body of a repeat.

`elements` (list of music objects)
: A list of elements for sequential of simultaneous music, or the alternatives of repeated music.

`elements-callback` (procedure)
: Return a list of children, for use by a sequential iterator. Takes a single music parameter.

`error-found` (boolean)
: If true, a parsing error was found in this expression.

`figure` (integer)
: A bass figure.

`footnote-text` (markup)
: Text to appear in a footnote.

`force-accidental` (boolean)
: If set, a cautionary accidental should always be printed on this note.

`grob-property` (symbol)
: The symbol of the grob property to set.

`grob-property-path` (list)
: A list of symbols, locating a nested grob property, e.g., `(beamed-lengths details)`.

`grob-value` (any type)
: The value of the grob property to set.

`id` (symbol)
: The ID of an event.

`input-tag` (any type)
: Arbitrary marker to relate input and output.

`inversion` (boolean)
: If set, this chord note is inverted.

`iterator-ctor` (procedure)
: Function to construct a `music-event-iterator` object for this music.

`label` (number or markup)
: Label of a mark.

`last-pitch` (pitch)
: The last pitch after relativization.

`length` (moment)
: The duration of this music.

`length-callback` (procedure)
: How to compute the duration of this music. This property can only be defined as initializer in `scm/define-music-types.scm`.

`line-break-permission` (symbol)
: When the music is at top-level, whether to allow, forbid or force a line break.

`metronome-count` (number or pair)
: How many beats in a minute?

`midi-extra-velocity` (integer)
: How much louder or softer should this note be in MIDI output? The default is 0.

`midi-length` (procedure)
: Function to determine how long to play a note in MIDI. It should take a moment (the written length of the note) and a context, and return a moment (the length to play the note).

`moment` (moment)
: The moment at which an event happens.

`music-cause` (music)
: The music object that is the cause of an event.

`name` (symbol)
: Name of this music object.

`no-continuation` (boolean)
: If set, disallow continuation lines.

`numerator` (integer)
: Numerator of a time signature.

`octavation` (integer)
: This pitch was octavated by how many octaves? For chord inversions, this is negative.

`once` (boolean)
: Apply this operation only during one time step?

`ops` (any type)
: The operations to apply during the creation of a context.

`origin` (input location)
: Where was this piece of music defined?

`ottava-number` (integer)
: The octavation for `\ottava`.

`page-break-permission` (symbol)
: When the music is at top-level, whether to allow, forbid or force a page break.

`page-label` (symbol)
: The label of a page marker.

`page-marker` (boolean)
: If true, and the music expression is found at top-level, a page marker object is instanciated instead of a score.

`page-turn-permission` (symbol)
: When the music is at top-level, whether to allow, forbid or force a page turn.

`parenthesize` (boolean)
: Enclose resulting objects in parentheses?

`part-combine-status` (symbol)
: Change to what kind of state? Options are `solo1`, `solo2` and `unisono`.

`pitch` (pitch)
: The pitch of this note.

`pitch-alist` (list)
: A list of pitches jointly forming the scale of a key signature.

`pop-first` (boolean)
: Do a revert before we try to do an override on some grob property.

`procedure` (procedure)
: The function to run with `\applycontext`. It must take a single argument, being the context.

`property-operations` (list)
: Do these operations for instantiating the context.

`property-path` (symbol)
: The path of a property.

`quoted-context-id` (string)
: The ID of the context to direct quotes to, e.g., `cue`.

`quoted-context-type` (symbol)
: The name of the context to direct quotes to, e.g., `Voice`.

`quoted-events` (vector)
: A vector of with `moment` and `event-list` entries.

`quoted-music-clef` (string)
: The clef of the voice to quote.

`quoted-music-name` (string)
: The name of the voice to quote.

`quoted-transposition` (pitch)
: The pitch used for the quote, overriding `\transposition`.

`quoted-voice-direction` (direction)
: Should the quoted voice be up-stem or down-stem?

`repeat-count` (integer)
: Do a `\repeat` how often?

`slash-count` (integer)
: The number of slashes in a single-beat repeat. If zero, signals a beat containing varying durations.

`span-direction` (direction)
: Does this start or stop a spanner?

`span-text` (markup)
: The displayed text for dynamic text spanners (e.g., cresc.)

`span-type` (symbol)
: What kind of dynamic spanner should be created? Options are `'text` and `'hairpin`.

`spanner-id` (index or symbol)
: Identifier to distinguish concurrent spanners.

`start-callback` (procedure)
: Function to compute the negative length of starting grace notes. This property can only be defined as initializer in `scm/define-music-types.scm`.

`string-number` (integer)
: The number of the string in a `StringNumberEvent`.

`symbol` (symbol)
: Grob name to perform an override or revert on.

`tags` (list)
: List of symbols that for denoting extra details, e.g., `\tag #'part ...` could tag a piece of music as only being active in a part.

`tempo-unit` (duration)
: The unit for the metronome count.

`text` (markup)
: Markup expression to be printed.

`to-relative-callback` (procedure)
: How to transform a piece of music to relative pitches.

`tonic` (pitch)
: Base of the scale.

`tremolo-type` (integer)
: Speed of tremolo, e.g., 16 for `c4:16`.

`trill-pitch` (pitch)
: Pitch of other note of the trill.

`tweaks` (list)
: An alist of properties to override in the backend for the grob made of this event.

`type` (symbol)
: The type of this music object. Determines iteration in some cases.

`types` (list)
: The types of this music object; determines by what engraver this music expression is processed.

`untransposable` (boolean)
: If set, this music is not transposed.

`value` (any type)
: Assignment value for a translation property.

`void` (boolean)
: If this property is `#t`, then the music expression is to be discarded by the toplevel music handler.

`volta-repeats` (list)
: A list that is transformed into a volta repeat element list.

`what` (symbol)
: What to change for auto-change.

  FIXME: Naming.

`X-offset` (number)
: Offset of resulting grob; only used for balloon texts.

`Y-offset` (number)
: Offset of resulting grob; only used for balloon texts.

2 Translation

2.1 Contexts

2.1.1 ChoirStaff

Identical to `StaffGroup` except that the contained staves are not connected vertically.

This context creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, and Section 3.1.136 [VerticalAlignment], page 528.

This context sets the following properties:

- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `shortVocalName` to `'()`.
- Set translator property `systemStartDelimiter` to `'SystemStartBracket`.
- Set translator property `topLevelAlignment` to `#f`.
- Set translator property `vocalName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context ChoirStaff can contain Section 2.1.1 [ChoirStaff], page 57, Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.11 [GrandStaff], page 101, Section 2.1.16 [Lyrics], page 151, Section 2.1.21 [OneStaff], page 183, Section 2.1.24 [PianoStaff], page 208, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, and Section 2.1.28 [StaffGroup], page 246.

This context is built from the following engraver(s):

`Section 2.2.55 [Instrument_name_engraver], page 328`

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.119 [System_start_delimiter_engraver], page 348`
: Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`Section 2.2.135 [Vertical_align_engraver], page 354`
: Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

2.1.2 ChordNames

Typesets chord names.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.24 [ChordName], page 396, Section 3.1.106 [StaffSpacing], page 492, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `font-size` in Section 3.1.85 [ParenthesesItem], page 471, to `1.5`.
- Set grob-property `nonstaff-nonstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `0.5`.
- Set grob-property `nonstaff-relatedstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `0.5`.
- Set grob-property `remove-empty` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `remove-first` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `staff-affinity` in Section 3.1.137 [VerticalAxisGroup], page 528, to `-1`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.15 [Chord_name_engraver], page 313`

Catch note and rest events and generate the appropriate chordname.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains (*`chord . markup`*) entries.

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains (*`chord . markup`*) entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`lastChord` (markup)
: Last chord, used for detecting chord changes.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

Properties (write)

`lastChord` (markup)
: Last chord, used for detecting chord changes.

This engraver creates the following layout object(s):

Section 3.1.24 [ChordName], page 396.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
: Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

2.1.3 CueVoice

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and subscripts, slurs, ties, and rests.

You have to instantiate this explicitly if you want to have multiple voices on the same staff.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `beam-thickness` in Section 3.1.19 [Beam], page 390, to `0.35`.
- Set grob-property `ignore-ambitus` in Section 3.1.80 [NoteHead], page 467, to `#t`.
- Set grob-property `length-fraction` in Section 3.1.19 [Beam], page 390, to `0.629960524947437`.
- Set grob-property `length-fraction` in Section 3.1.109 [Stem], page 494, to `0.629960524947437`.
- Set translator property `fontSize` to `-4`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`

Generate an Arpeggio symbol.

Music types accepted:

Section 1.2.5 [arpeggio-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`

Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`
Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`

Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.63 [Ligature_bracket_engraver], page 331`
Handle `Ligature_events` by engraving `Ligature` brackets.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.64 [LigatureBracket], page 449.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`
: Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

  Properties (read)

  `fingeringOrientations` (list)
  : A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

  `harmonicDots` (boolean)
  : If set, harmonic notes in dotted chords get dots.

  `stringNumberOrientations` (list)
  : See `fingeringOrientations`.

  `strokeFingerOrientations` (list)
  : See `fingeringOrientations`.

  This engraver creates the following layout object(s):

  Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
: Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set ‘Solo’ and ‘A due’ texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice ‘two’ when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
: Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
: Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
: Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
: Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`

Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.4 Devnull

Silently discards all musical information given to this context.

This context also accepts commands for the following context(s):

Staff and Voice.

This context creates the following layout object(s):

none.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

2.1.5 DrumStaff

Handles typesetting for percussion.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `staff-padding` in Section 3.1.96 [Script], page 483, to `0.75`.
- Set translator property `clefGlyph` to `"clefs.percussion"`.

- Set translator property `clefPosition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.6 [DrumVoice], page 81.

Context DrumStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, and Section 2.1.20 [NullVoice], page 180.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.27 [Dot_column_engraver], page 318`
: Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
: Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.62 [Ledger_line_engraver], page 331`
Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.6 DrumVoice

A voice on a percussion staff.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.44 [Flag], page 425, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.4 [Auto_beam_engraver], page 308`
: Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

  Music types accepted:

  Section 1.2.9 [beam-forbid-event], page 42,

  Properties (read)

  `autoBeaming` (boolean)
  : If set to true then beams are generated automatically.

  `baseMoment` (moment)
  : Smallest unit of time that will stand on its own as a subdivided section.

  `beamExceptions` (list)
  : An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`

Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`

Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`

Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.28 [Dots_engraver], page 319`

Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`

Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.31 [Drum_notes_engraver], page 320`

Generate drum note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`',

‘agostini-drums-style’, ‘timbales-style’, ‘congas-style’, ‘bongos-style’, and ‘percussion-style’.

The layout style is a hash table, containing the drum-pitches (e.g., the symbol ‘hihat’) as keys, and a list *(notehead-style script vertical-position)* as values.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467, and Section 3.1.96 [Script], page 483.

Section 2.2.32 [Dynamic_align_engraver], page 320

Align hairpins and dynamic texts on a horizontal line.

Properties (read)

currentMusicalColumn (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

Section 2.2.33 [Dynamic_engraver], page 320

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

crescendoSpanner (symbol)

The type of spanner to be used for crescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin crescendo is used.

crescendoText (markup)

The text to print at start of non-hairpin crescendo, i.e., ‘cresc.’.

currentMusicalColumn (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

decrescendoSpanner (symbol)

The type of spanner to be used for decrescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin decrescendo is used.

decrescendoText (markup)

The text to print at start of non-hairpin decrescendo, i.e., ‘dim.’.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
If set to `#t`, prevent a line break at this point.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`
Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
Signal if a beam is present.

`beatStructure` (list)
List of `baseMoments` that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`

Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`

Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
: Create text scripts.

  Music types accepted:

  Section 1.2.69 [text-script-event], page 50,

  This engraver creates the following layout object(s):

  Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`
: Create text spanner from an event.

  Music types accepted:

  Section 1.2.70 [text-span-event], page 50,

  Properties (read)

  `currentMusicalColumn` (graphical (layout) object)
  : Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

  This engraver creates the following layout object(s):

  Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`
: Generate ties between note heads of equal pitch.

  Music types accepted:

  Section 1.2.71 [tie-event], page 50,

  Properties (read)

  `skipTypesetting` (boolean)
  : If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

  `tieWaitForNote` (boolean)
  : If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`
Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`
Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.7 Dynamics

Holds a single line of dynamics, which will be centered between the staves surrounding this context.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.11 [BarLine], page 381, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420,

Section 3.1.52 [Hairpin], page 432, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.96 [Script], page 483, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.133 [UnaCordaPedal], page 525, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `font-shape` in Section 3.1.122 [TextScript], page 510, to `'italic`.
- Set grob-property `nonstaff-relatedstaff-spacing` in Section 3.1.137 [VerticalAxisGroup], page 528, to:

  ```
  '((basic-distance . 5) (padding . 0.5))
  ```
- Set grob-property `outside-staff-priority` in Section 3.1.38 [DynamicLineSpanner], page 417, to `#f`.
- Set grob-property `outside-staff-priority` in Section 3.1.39 [DynamicText], page 418, to `#f`.
- Set grob-property `outside-staff-priority` in Section 3.1.52 [Hairpin], page 432, to `#f`.
- Set grob-property `staff-affinity` in Section 3.1.137 [VerticalAxisGroup], page 528, to `0`.
- Set grob-property `Y-offset` in Section 3.1.38 [DynamicLineSpanner], page 417, to `0`.
- Set translator property `pedalSustainStrings` to:

  ```
  '("Ped." "*Ped." "*")
  ```
- Set translator property `pedalUnaCordaStrings` to:

  ```
  '("una corda" "" "tre corde")
  ```

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.32 [Dynamic_align_engraver], page 320`

Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`
Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

2.1.8 FiguredBass

A context for printing a figured bass line.

This context creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.106 [StaffSpacing], page 492, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `nonstaff-nonstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `0.5`.
- Set grob-property `nonstaff-relatedstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `0.5`.
- Set grob-property `remove-empty` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `remove-first` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `staff-affinity` in Section 3.1.137 [VerticalAxisGroup], page 528, to `1`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
Don't swallow rest events.

`implicitBassFigures` (list)
A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
: Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

2.1.9 FretBoards

A context for displaying fret diagrams.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.47 [FretBoard], page 428, Section 3.1.55 [InstrumentName], page 436, Section 3.1.106 [StaffSpacing], page 492, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set translator property `handleNegativeFrets` to `'recalculate`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `predefinedDiagramTable` to `#<hash-table 0/113>`.
- Set translator property `restrainOpenStrings` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`
: Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.44 [Fretboard_engraver], page 324`
Generate fret diagram from one or more events of type `NoteEvent`.

Music types accepted:

Section 1.2.23 [fingering-event], page 44, Section 1.2.41 [note-event], page 46, and Section 1.2.65 [string-number-event], page 49,

Properties (read)

`chordChanges` (boolean)
: Only show changes in chords scheme?

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list

of tabstring events, and the fretboard grob if a fretboard is desired.

`predefinedDiagramTable` (hash table)
The hash table of predefined fret diagrams to use in FretBoards.

`stringTunings` (list)
The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`tablatureFormat` (procedure)
A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

This engraver creates the following layout object(s):

Section 3.1.47 [FretBoard], page 428.

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
See `instrumentName`.

`shortVocalName` (markup)
Name of a vocal line, short version.

`vocalName` (markup)
Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

2.1.10 Global

Hard coded entry point for LilyPond. Cannot be tuned.

This context creates the following layout object(s):

none.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.26 [Score], page 214.

Context Global can contain Section 2.1.26 [Score], page 214.

2.1.11 GrandStaff

A group of staves, with a brace on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.55 [InstrumentName], page 436, Section 3.1.103 [SpanBar], page 490, Section 3.1.104 [SpanBarStub], page 491, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, and Section 3.1.136 [VerticalAlignment], page 528.

This context sets the following properties:

- Set grob-property `extra-spacing-width` in Section 3.1.39 [DynamicText], page 418, to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `systemStartDelimiter` to `'SystemStartBrace`.
- Set translator property `topLevelAlignment` to `#f`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context GrandStaff can contain Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.16 [Lyrics], page 151, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, and Section 2.1.29 [TabStaff], page 248.

This context is built from the following engraver(s):

`Section 2.2.55 [Instrument_name_engraver], page 328`
: Create a system start text for instrument or vocal names.

  Properties (read)

  `currentCommandColumn` (graphical (layout) object)
  : Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.108 [Span_arpeggio_engraver], page 346`
: Make arpeggios that span multiple staves.

Properties (read)

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.109 [Span_bar_engraver], page 346`
: Make cross-staff bar lines: It catches all normal bar lines and draws a single span bar across them.

This engraver creates the following layout object(s):

Section 3.1.103 [SpanBar], page 490.

`Section 2.2.110 [Span_bar_stub_engraver], page 346`
: Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s):

Section 3.1.104 [SpanBarStub], page 491.

`Section 2.2.119 [System_start_delimiter_engraver], page 348`
: Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`Section 2.2.135 [Vertical_align_engraver], page 354`

Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

2.1.12 GregorianTranscriptionStaff

Handles clefs, bar lines, keys, accidentals. It can contain `Voice` contexts.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `transparent` in Section 3.1.11 [BarLine], page 381, to `#t`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.

- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.13 [GregorianTranscriptionVoice], page 114.

Context GregorianTranscriptionStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, and Section 2.1.20 [NullVoice], page 180.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

Properties (read)

`accidentalGrouping` (symbol)

If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`autoAccidentals` (list)

List of different ways to typeset an accidental.

For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

Each entry in the list is either a symbol or a procedure.

symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos`
: The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`
: Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`
: Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.27 [Dot_column_engraver], page 318`
Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigure-Alignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
See `instrumentName`.

`shortVocalName` (markup)
Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`
Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`*` . `*`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
See `pedalSustainStyle`.

`pedalSustainStrings` (list)
A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
A queue of `(end-moment . grob)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.13 GregorianTranscriptionVoice

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and subscripts, slurs, ties, and rests.

You have to instantiate this explicitly if you want to have multiple voices on the same staff.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `padding` in Section 3.1.96 [Script], page 483, to `0.5`.
- Set grob-property `transparent` in Section 3.1.64 [LigatureBracket], page 449, to `#t`.
- Set translator property `autoBeaming` to `#f`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`

Generate an Arpeggio symbol.

Music types accepted:

Section 1.2.5 [arpeggio-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`

Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.35 [Episema_engraver], page 321`
Create an *Editio Vaticana*-style episema line.

Music types accepted:

Section 1.2.21 [episema-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.41 [Episema], page 422.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`
Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`
: Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`
: Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`
Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.63 [Ligature_bracket_engraver], page 331`
Handle `Ligature_events` by engraving `Ligature` brackets.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.64 [LigatureBracket], page 449.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`

Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

Properties (read)

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
: Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
: Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
: Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
: Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`

Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`

Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`

Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
: The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`

Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
See `stemLeftBeamCount`.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`

Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)

If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)

If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)

Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)

Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.14 KievanStaff

Same as `Staff` context, except that it is accommodated for typesetting a piece in Kievan style.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set translator property `autoAccidentals` to:

```
'(Staff #<procedure #f (context pitch barnum measurepos)>
        #<procedure neo-modern-accidental-rule (context pitch barnum measurepos)>)
```

- Set translator property `autoCautionaries` to `'()`.
- Set translator property `clefGlyph` to `"clefs.kievan.do"`.

- Set translator property `clefPosition` to `0`.
- Set translator property `clefTransposition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `extraNatural` to `#f`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `middleCClefPosition` to `0`.
- Set translator property `middleCPosition` to `0`.
- Set translator property `printKeyCancellation` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.15 [KievanVoice], page 137.

Context KievanStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.15 [KievanVoice], page 137, and Section 2.1.20 [NullVoice], page 180.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`
: Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

  Properties (read)

  `accidentalGrouping` (symbol)
  : If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

  `autoAccidentals` (list)
  : List of different ways to typeset an accidental.

    For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

    Each entry in the list is either a symbol or a procedure.

    symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

    procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`
Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.27 [Dot_column_engraver], page 318`
: Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
: Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`

Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`* `.` *`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`* `.` *`alter`*`)` or `((`*`octave`* `.` *`step`*`)` `.` *`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
: Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
: Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
See `pedalSustainStyle`.

`pedalSustainStrings` (list)
A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

2.1.15 KievanVoice

Same as `Voice` context, except that it is accommodated for typesetting a piece in Kievan style.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.59 [KievanLigature], page 444, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `duration-log` in Section 3.1.80 [NoteHead], page 467, to `note-head::calc-kievan-duration-log`.
- Set grob-property `glyph-name-alist` in Section 3.1.1 [Accidental], page 370, to:

```
'((-1/2 . "accidentals.kievanM1")
  (1/2 . "accidentals.kievan1"))
```

- Set grob-property `length` in Section 3.1.109 [Stem], page 494, to `0.0`.
- Set grob-property `positions` in Section 3.1.19 [Beam], page 390, to `beam::get-kievan-positions`.
- Set grob-property `quantized-positions` in Section 3.1.19 [Beam], page 390, to `beam::get-kievan-quantized-positions`.
- Set grob-property `stencil` in Section 3.1.44 [Flag], page 425, to `#f`.
- Set grob-property `stencil` in Section 3.1.99 [Slur], page 485, to `#f`.
- Set grob-property `stencil` in Section 3.1.109 [Stem], page 494, to `#f`.
- Set grob-property `style` in Section 3.1.34 [Dots], page 412, to `'kievan`.
- Set grob-property `style` in Section 3.1.80 [NoteHead], page 467, to `'kievan`.
- Set grob-property `style` in Section 3.1.94 [Rest], page 481, to `'mensural`.
- Set grob-property `X-offset` in Section 3.1.109 [Stem], page 494, to `stem::kievan-offset-callback`.
- Set translator property `autoBeaming` to `#f`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`
: Generate an Arpeggio symbol.

  Music types accepted:

  Section 1.2.5 [arpeggio-event], page 42,

  This engraver creates the following layout object(s):

  Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`
: Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

  Music types accepted:

  Section 1.2.9 [beam-forbid-event], page 42,

  Properties (read)

  `autoBeaming` (boolean)
  : If set to true then beams are generated automatically.

  `baseMoment` (moment)
  : Smallest unit of time that will stand on its own as a subdivided section.

  `beamExceptions` (list)
  : An alist of exceptions to autobeam rules that normally end on beats.

  `beamHalfMeasure` (boolean)
  : Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

  `beatStructure` (list)
  : List of `baseMoment`s that are combined to make beats.

  `subdivideBeams` (boolean)
  : If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

  This engraver creates the following layout object(s):

  Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`
: Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

  Music types accepted:

  Section 1.2.8 [beam-event], page 42,

  Properties (read)

  `baseMoment` (moment)
  : Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
Signal if a beam is present.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.40 [Fingering_engraver], page 323`

Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`

Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns.

The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`

Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.60 [Kievan_ligature_engraver], page 330`
Handle `Kievan_ligature_events` by glueing Kievan heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.59 [KievanLigature], page 444.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`
: Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

  Properties (read)

  `fingeringOrientations` (list)
  : A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

  `harmonicDots` (boolean)
  : If set, harmonic notes in dotted chords get dots.

  `stringNumberOrientations` (list)
  : See `fingeringOrientations`.

  `strokeFingerOrientations` (list)
  : See `fingeringOrientations`.

  This engraver creates the following layout object(s):

  Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
Set ‘Solo’ and ‘A due’ texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
See `stemLeftBeamCount`.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`

Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)

If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)

If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)

Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)

Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.16 Lyrics

Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

This context creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436, Section 3.1.65 [LyricExtender], page 450, Section 3.1.66 [LyricHyphen], page 451, Section 3.1.67 [LyricSpace], page 452, Section 3.1.68 [LyricText], page 453, Section 3.1.108 [StanzaNumber], page 493, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `bar-extent` in Section 3.1.11 [BarLine], page 381, to:
  ```
  '(-0.05 . 0.05)
  ```
- Set grob-property `font-size` in Section 3.1.55 [InstrumentName], page 436, to `1.0`.
- Set grob-property `nonstaff-nonstaff-spacing` in Section 3.1.137 [VerticalAxisGroup], page 528, to:
  ```
  '((basic-distance . 0)
    (minimum-distance . 2.8)
    (padding . 0.2)
    (stretchability . 0))
  ```
- Set grob-property `nonstaff-relatedstaff-spacing` in Section 3.1.137 [VerticalAxisGroup], page 528, to:
  ```
  '((basic-distance . 5.5)
    (padding . 0.5)
    (stretchability . 1))
  ```
- Set grob-property `nonstaff-unrelatedstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `1.5`.
- Set grob-property `remove-empty` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `remove-first` in Section 3.1.137 [VerticalAxisGroup], page 528, to `#t`.
- Set grob-property `self-alignment-Y` in Section 3.1.55 [InstrumentName], page 436, to `#f`.
- Set grob-property `staff-affinity` in Section 3.1.137 [VerticalAxisGroup], page 528, to `1`.

- Set translator property `instrumentName` to `'()`.
- Set translator property `searchForVoice` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`
: Group all objects created in this context in a `VerticalAxisGroup` spanner.

  Properties (read)

  `currentCommandColumn` (graphical (layout) object)
  : Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

  `hasAxisGroup` (boolean)
  : True if the current context is contained in an axis group.

  `keepAliveInterfaces` (list)
  : A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

  Properties (write)

  `hasAxisGroup` (boolean)
  : True if the current context is contained in an axis group.

  This engraver creates the following layout object(s):

  Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.36 [Extender_engraver], page 322`
: Create lyric extenders.

  Music types accepted:

  Section 1.2.16 [completize-extender-event], page 43, and Section 1.2.22 [extender-event], page 43,

  Properties (read)

  `extendersOverRests` (boolean)
  : Whether to continue extenders as they cross a rest.

  This engraver creates the following layout object(s):

  Section 3.1.65 [LyricExtender], page 450.

`Section 2.2.41 [Font_size_engraver], page 323`
: Put `fontSize` into `font-size` grob property.

  Properties (read)

  `fontSize` (number)
  : The relative size of all grobs in a context.

`Section 2.2.54 [Hyphen_engraver], page 328`
Create lyric hyphens and distance constraints between words.

Music types accepted:

Section 1.2.27 [hyphen-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.66 [LyricHyphen], page 451, and Section 3.1.67 [LyricSpace], page 452.

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
See `instrumentName`.

`shortVocalName` (markup)
Name of a vocal line, short version.

`vocalName` (markup)
Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.64 [Lyric_engraver], page 331`
Engrave text for lyrics.

Music types accepted:

Section 1.2.34 [lyric-event], page 45,

Properties (read)

`ignoreMelismata` (boolean)
Ignore melismata for this Section "Lyrics" in *Internals Reference* line.

`lyricMelismaAlignment` (number)
Alignment to use for a melisma syllable.

`searchForVoice` (boolean)
Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

This engraver creates the following layout object(s):

Section 3.1.68 [LyricText], page 453.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
: Coordinates items that get their pure heights from their neighbors.

`Section 2.2.117 [Stanza_number_engraver], page 348`
: Engrave stanza numbers.

  Properties (read)

  `stanza` (markup)
  : Stanza 'number' to print before the start of a verse. Use in `Lyrics` context.

  This engraver creates the following layout object(s):

  Section 3.1.108 [StanzaNumber], page 493.

2.1.17 MensuralStaff

Same as `Staff` context, except that it is accommodated for typesetting a piece in mensural style.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `glyph-name-alist` in Section 3.1.4 [AccidentalSuggestion], page 373, to:

```
'((-1/2 . "accidentals.mensuralM1")
  (0 . "accidentals.vaticana0")
  (1/2 . "accidentals.mensural1"))
```

- Set grob-property `glyph-name-alist` in Section 3.1.1 [Accidental], page 370, to:

```
'((-1/2 . "accidentals.mensuralM1")
  (0 . "accidentals.vaticana0")
  (1/2 . "accidentals.mensural1"))
```

- Set grob-property `glyph-name-alist` in Section 3.1.58 [KeySignature], page 441, to:

```
'((-1/2 . "accidentals.mensuralM1")
  (0 . "accidentals.vaticana0")
  (1/2 . "accidentals.mensural1"))
```

- Set grob-property `neutral-direction` in Section 3.1.32 [Custos], page 410, to `-1`.
- Set grob-property `neutral-position` in Section 3.1.32 [Custos], page 410, to `3`.
- Set grob-property `style` in Section 3.1.32 [Custos], page 410, to `'mensural`.
- Set grob-property `style` in Section 3.1.126 [TimeSignature], page 515, to `'mensural`.
- Set grob-property `thickness` in Section 3.1.107 [StaffSymbol], page 493, to `0.6`.
- Set grob-property `transparent` in Section 3.1.11 [BarLine], page 381, to `#t`.
- Set translator property `autoAccidentals` to:
  `'(Staff #<procedure #f (context pitch barnum measurepos)>)`
- Set translator property `autoCautionaries` to `'()`.
- Set translator property `clefGlyph` to `"clefs.mensural.g"`.
- Set translator property `clefPosition` to `-2`.
- Set translator property `clefTransposition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `extraNatural` to `#f`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `middleCClefPosition` to `-6`.
- Set translator property `middleCPosition` to `-6`.
- Set translator property `printKeyCancellation` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.18 [MensuralVoice], page 165.

Context MensuralStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.18 [MensuralVoice], page 165, and Section 2.1.20 [NullVoice], page 180.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`
: Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

  Properties (read)

  `accidentalGrouping` (symbol)
  : If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

  `autoAccidentals` (list)
  : List of different ways to typeset an accidental.

    For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

    Each entry in the list is either a symbol or a procedure.

    symbol The symbol is the name of the context in which the following rules are

to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` .

step) . *alter*), where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain ((*octave* . *name*) . (*alter barnumber* . *measureposition*)) pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain ((*octave* . *name*) . (*alter barnumber* . *measureposition*)) pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`
Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`
Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
Name of the symbol within the music font.

`cueClefPosition` (number)
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
'`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.25 [Custos_engraver], page 317`
Engrave custodes.

This engraver creates the following layout object(s):

Section 3.1.32 [Custos], page 410.

`Section 2.2.27 [Dot_column_engraver], page 318`
Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(end-moment . grob)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`

Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`*` . `*`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
: Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
: Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
: Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
: Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
: Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
: Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
: Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.18 MensuralVoice

Same as `Voice` context, except that it is accommodated for typesetting a piece in mensural style.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.72 [MensuralLigature], page 457, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87

[PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `style` in Section 3.1.44 [Flag], page 425, to `'mensural`.
- Set grob-property `style` in Section 3.1.80 [NoteHead], page 467, to `'mensural`.
- Set grob-property `style` in Section 3.1.94 [Rest], page 481, to `'mensural`.
- Set translator property `autoBeaming` to `#f`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`
: Generate an Arpeggio symbol.

  Music types accepted:

  Section 1.2.5 [arpeggio-event], page 42,

  This engraver creates the following layout object(s):

  Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`
: Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

  Music types accepted:

  Section 1.2.9 [beam-forbid-event], page 42,

  Properties (read)

  `autoBeaming` (boolean)
  : If set to true then beams are generated automatically.

  `baseMoment` (moment)
  : Smallest unit of time that will stand on its own as a subdivided section.

  `beamExceptions` (list)
  : An alist of exceptions to autobeam rules that normally end on beats.

  `beamHalfMeasure` (boolean)
  : Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`

Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`

Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*`end-moment . grob`*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`
Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.70 [Mensural_ligature_engraver], page 333`
Handle `Mensural_ligature_events` by glueing special ligature heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.72 [MensuralLigature], page 457.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`
Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

Properties (read)

`fingeringOrientations` (list)
A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`harmonicDots` (boolean)
If set, harmonic notes in dotted chords get dots.

`stringNumberOrientations` (list)
See `fingeringOrientations`.

`strokeFingerOrientations` (list)
See `fingeringOrientations`.

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`

Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`

Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
See `stemLeftBeamCount`.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`
Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`
Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.19 NoteNames

A context for printing the names of notes.

This context creates the following layout object(s):

Section 3.1.81 [NoteName], page 468, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `nonstaff-nonstaff-spacing` in Section 3.1.137 [VerticalAxisGroup], page 528, to:

```
'((basic-distance . 0)
  (minimum-distance . 2.8)
  (padding . 0.2)
  (stretchability . 0))
```

- Set grob-property `nonstaff-relatedstaff-spacing` in Section 3.1.137 [VerticalAxisGroup], page 528, to:

```
'((basic-distance . 5.5)
  (padding . 0.5)
  (stretchability . 1))
```

- Set grob-property `nonstaff-unrelatedstaff-spacing.padding` in Section 3.1.137 [VerticalAxisGroup], page 528, to `1.5`.
- Set grob-property `staff-affinity` in Section 3.1.137 [VerticalAxisGroup], page 528, to `1`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.78 [Note_name_engraver], page 336`
Print pitches as words.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`printOctaveNames` (boolean)
Print octave marks for the `NoteNames` context.

This engraver creates the following layout object(s):

Section 3.1.81 [NoteName], page 468.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

2.1.20 NullVoice

For aligning lyrics without printing notes

This context also accepts commands for the following context(s):

Staff and Voice.

This context creates the following layout object(s):

Section 3.1.19 [Beam], page 390, Section 3.1.80 [NoteHead], page 467, Section 3.1.99 [Slur], page 485, Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

This context sets the following properties:

- Set grob-property `no-ledgers` in Section 3.1.80 [NoteHead], page 467, to `#t`.
- Set grob-property `stencil` in Section 3.1.19 [Beam], page 390, to `#f`.
- Set grob-property `stencil` in Section 3.1.80 [NoteHead], page 467, to `#f`.
- Set grob-property `stencil` in Section 3.1.99 [Slur], page 485, to `#f`.
- Set grob-property `stencil` in Section 3.1.124 [Tie], page 513, to `#f`.
- Set grob-property `X-extent` in Section 3.1.80 [NoteHead], page 467, to `#<procedure #f (g)>`.
- Set translator property `nullAccidentals` to `#t`.
- Set translator property `squashedPosition` to `0`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.10 [Beam_engraver], page 312`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.77 [Note_heads_engraver], page 336`

Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.92 [Pitch_squash_engraver], page 341`
Set the vertical position of note heads to `squashedPosition`, if that property is set. This can be used to make a single-line staff demonstrating the rhythm of a melody.

Properties (read)

`squashedPosition` (integer)
Vertical position of squashing for Section "Pitch_squash_engraver" in *Internals Reference*.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

2.1.21 OneStaff

Provides a common axis for the contained staves, making all of them appear in the same vertical space. This can be useful for typesetting staves of different types in immediate succession or for temporarily changing the character of one staff or overlaying it with a different one. Often used with `\stopStaff` and `\startStaff` for best results.

This context creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context OneStaff can contain Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.9 [FretBoards], page 98, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.19 [NoteNames], page 178, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

2.1.22 PetrucciStaff

Same as `Staff` context, except that it is accommodated for typesetting a piece in Petrucci style.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373,

Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `neutral-direction` in Section 3.1.32 [Custos], page 410, to `-1`.
- Set grob-property `neutral-position` in Section 3.1.32 [Custos], page 410, to `3`.
- Set grob-property `style` in Section 3.1.32 [Custos], page 410, to `'mensural`.
- Set grob-property `thickness` in Section 3.1.107 [StaffSymbol], page 493, to `1.3`.
- Set translator property `autoAccidentals` to:

```
'(Staff #<procedure #f (context pitch barnum measurepos)>
        #<procedure neo-modern-accidental-rule (context pitch barnum measurepos)>)
```

- Set translator property `autoCautionaries` to `'()`.
- Set translator property `clefGlyph` to `"clefs.petrucci.g"`.
- Set translator property `clefPosition` to `-2`.
- Set translator property `clefTransposition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `extraNatural` to `#f`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `middleCClefPosition` to `-6`.
- Set translator property `middleCPosition` to `-6`.
- Set translator property `printKeyCancellation` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.23 [PetrucciVoice], page 194.

Context PetrucciStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.20 [NullVoice], page 180, and Section 2.1.23 [PetrucciVoice], page 194.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`
: Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

Properties (read)

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol
  : The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

  procedure
  : The procedure represents an accidental rule to be applied to the previously specified context.

    The procedure takes the following arguments:

    `context`
    : The current context to which the rule should be applied.

    `pitch`
    : The pitch of the note to be evaluated.

    `barnum`
    : The current bar number.

    `measurepos`
    : The current measure position.

    The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`
: Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`
: Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.25 [Custos_engraver], page 317`

Engrave custodes.

This engraver creates the following layout object(s):

Section 3.1.32 [Custos], page 410.

`Section 2.2.27 [Dot_column_engraver], page 318`
Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`

Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`*` . `*`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
: Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
: Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(end-moment . grob)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
: Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
: Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.23 PetrucciVoice

Same as `Voice` context, except that it is accommodated for typesetting a piece in Petrucci style.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414,

Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.72 [MensuralLigature], page 457, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `length` in Section 3.1.109 [Stem], page 494, to `5`.
- Set grob-property `style` in Section 3.1.80 [NoteHead], page 467, to `'petrucci`.
- Set grob-property `style` in Section 3.1.94 [Rest], page 481, to `'mensural`.
- Set grob-property `thickness` in Section 3.1.109 [Stem], page 494, to `1.7`.
- Set translator property `autoBeaming` to `#f`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`

Generate an Arpeggio symbol.

Music types accepted:

Section 1.2.5 [arpeggio-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`

Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)

If set to true then beams are generated automatically.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`
: Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
: Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`

Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`

Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment`*` . `*`grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment`*` . `*`grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.70 [Mensural_ligature_engraver], page 333`
Handle `Mensural_ligature_events` by glueing special ligature heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.72 [MensuralLigature], page 457.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`
: Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

  Properties (read)

  `fingeringOrientations` (list)
  : A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

  `harmonicDots` (boolean)
  : If set, harmonic notes in dotted chords get dots.

  `stringNumberOrientations` (list)
  : See `fingeringOrientations`.

  `strokeFingerOrientations` (list)
  : See `fingeringOrientations`.

  This engraver creates the following layout object(s):

  Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
: Engrave a line between two note heads in a staff switch if `followVoice` is set.

  Properties (read)

  `followVoice` (boolean)
  : If set, note heads are tracked across staff switches by a thin line.

  This engraver creates the following layout object(s):

  Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
: Generate note heads.

  Music types accepted:

  Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
: Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
: Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
: Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
: Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
See `stemLeftBeamCount`.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`
Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.24 PianoStaff

Just like `GrandStaff`, but the staves are only removed together, never separately.

This context also accepts commands for the following context(s):

GrandStaff.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.55 [InstrumentName], page 436, Section 3.1.103 [SpanBar], page 490, Section 3.1.104 [SpanBarStub], page 491, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, and Section 3.1.136 [VerticalAlignment], page 528.

This context sets the following properties:

- Set grob-property `extra-spacing-width` in Section 3.1.39 [DynamicText], page 418, to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `systemStartDelimiter` to `'SystemStartBrace`.
- Set translator property `topLevelAlignment` to `#f`.
- Set translator property `topLevelAlignment` to `#f`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context PianoStaff can contain Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.16 [Lyrics], page 151, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, and Section 2.1.29 [TabStaff], page 248.

This context is built from the following engraver(s):

`Section 2.2.55 [Instrument_name_engraver], page 328`

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.57 [Keep_alive_together_engraver], page 329`
: This engraver collects all `Hara_kiri_group_spanner`s that are created in contexts at or below its own. These spanners are then tied together so that one will be removed only if all are removed. For example, if a `StaffGroup` uses this engraver, then the staves in the group will all be visible as long as there is a note in at least one of them.

`Section 2.2.108 [Span_arpeggio_engraver], page 346`
: Make arpeggios that span multiple staves.

  Properties (read)

  `connectArpeggios` (boolean)
  : If set, connect arpeggios across piano staff.

  This engraver creates the following layout object(s):

  Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.109 [Span_bar_engraver], page 346`
: Make cross-staff bar lines: It catches all normal bar lines and draws a single span bar across them.

  This engraver creates the following layout object(s):

  Section 3.1.103 [SpanBar], page 490.

`Section 2.2.110 [Span_bar_stub_engraver], page 346`
: Make stubs for span bars in all contexts that the span bars cross.

  This engraver creates the following layout object(s):

  Section 3.1.104 [SpanBarStub], page 491.

`Section 2.2.119 [System_start_delimiter_engraver], page 348`
: Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

  Properties (read)

  `currentCommandColumn` (graphical (layout) object)
  : Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

  `systemStartDelimiter` (symbol)
  : Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

  `systemStartDelimiterHierarchy` (pair)
  : A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`Section 2.2.135 [Vertical_align_engraver], page 354`
Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

`Section 2.2.135 [Vertical_align_engraver], page 354`
Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

2.1.25 RhythmicStaff

A context like `Staff` but for printing rhythms. Pitches are ignored; the notes are printed on one line.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.11 [BarLine], page 381, Section 3.1.33 [DotColumn], page 412, Section 3.1.55 [InstrumentName], page 436, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.126 [TimeSignature], page 515, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `line-count` in Section 3.1.107 [StaffSymbol], page 493, to `1`.

- Set grob-property `neutral-direction` in Section 3.1.19 [Beam], page 390, to `1`.
- Set grob-property `neutral-direction` in Section 3.1.109 [Stem], page 494, to `1`.
- Set grob-property `staff-padding` in Section 3.1.139 [VoltaBracket], page 531, to `3`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `squashedPosition` to `0`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.33 [Voice], page 293.

Context RhythmicStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.20 [NullVoice], page 180, and Section 2.1.33 [Voice], page 293.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.27 [Dot_column_engraver], page 318`

Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.41 [Font_size_engraver], page 323`

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.55 [Instrument_name_engraver], page 328`

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.62 [Ledger_line_engraver], page 331`

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.92 [Pitch_squash_engraver], page 341`
Set the vertical position of note heads to `squashedPosition`, if that property is set. This can be used to make a single-line staff demonstrating the rhythm of a melody.

Properties (read)

`squashedPosition` (integer)
Vertical position of squashing for Section "Pitch_squash_engraver" in *Internals Reference*.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.26 Score

This is the top level notation context. No other context can contain a `Score` context. This context handles the administration of time signatures. It also makes sure that items such as clefs, time signatures, and key-signatures are aligned across staves.

You cannot explicitly instantiate a `Score` context (since it is not contained in any other context). It is instantiated automatically when an output definition (a `\score` or `\layout` block) is processed.

This context also accepts commands for the following context(s):

Timing.

This context creates the following layout object(s):

Section 3.1.12 [BarNumber], page 385, Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, Section 3.1.45 [FootnoteItem], page 426, Section 3.1.46 [FootnoteSpanner], page 427, Section 3.1.49 [GraceSpacing], page 431, Section 3.1.63 [LeftEdge], page 447, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.84 [PaperColumn], page 470, Section 3.1.85 [ParenthesesItem], page 471, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.102 [SpacingSpanner], page 489, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, Section 3.1.136 [VerticalAlignment], page 528, Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

This context sets the following properties:

- Set translator property `additionalPitchPrefix` to `""`.
- Set translator property `aDueText` to `"a2"`.
- Set translator property `alternativeRestores` to:

  ```
  '(measurePosition measureLength lastChord)
  ```
- Set translator property `associatedVoiceType` to `'Voice`.
- Set translator property `autoAccidentals` to:

  ```
  '(Staff #<procedure #f (context pitch barnum measurepos)>)
  ```
- Set translator property `autoBeamCheck` to `default-auto-beam-check`.
- Set translator property `autoBeaming` to `#t`.
- Set translator property `autoCautionaries` to `'()`.
- Set translator property `automaticBars` to `#t`.
- Set translator property `barCheckSynchronize` to `#f`.
- Set translator property `barNumberFormatter` to `robust-bar-number-function`.
- Set translator property `barNumberVisibility` to `first-bar-number-invisible-and-no-parenthesized-bar-numbers`.
- Set translator property `beamHalfMeasure` to `#t`.
- Set translator property `chordNameExceptionsFull` to:

  ```
  '(((#<Pitch c' > #<Pitch e' > #<Pitch gis' >)
     (#<procedure line-markup (layout props args)>
      ("+")))
    ((#<Pitch c' > #<Pitch ees' > #<Pitch ges' >)
     (#<procedure line-markup (layout props args)>
      ((#<procedure super-markup (layout props arg)>
        "o"))))
    ((#<Pitch c' >
  ```

```
    #<Pitch ees' >
    #<Pitch ges' >
    #<Pitch bes' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure normal-size-super-markup (layout props arg)>
      "ø"))))
  ((#<Pitch c' >
    #<Pitch ees' >
    #<Pitch ges' >
    #<Pitch beses' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure super-markup (layout props arg)>
      "o7"))))
  ((#<Pitch c' >
    #<Pitch e' >
    #<Pitch g' >
    #<Pitch b' >
    #<Pitch fis'' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure super-markup (layout props arg)>
      "lyd"))))
  ((#<Pitch c' >
    #<Pitch e' >
    #<Pitch g' >
    #<Pitch bes' >
    #<Pitch des'' >
    #<Pitch ees'' >
    #<Pitch fis'' >
    #<Pitch aes'' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure super-markup (layout props arg)>
      "alt")))))
```

- Set translator property `chordNameExceptionsPartial` to:

```
'(((#<Pitch c' > #<Pitch d' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure normal-size-super-markup (layout props arg)>
      "2"))))
  ((#<Pitch c' > #<Pitch ees' >)
   (#<procedure line-markup (layout props args)>
    ("m")))
  ((#<Pitch c' > #<Pitch f' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure normal-size-super-markup (layout props arg)>
      "sus4"))))
  ((#<Pitch c' > #<Pitch g' >)
   (#<procedure line-markup (layout props args)>
    ((#<procedure normal-size-super-markup (layout props arg)>
      "5"))))
  ((#<Pitch c' > #<Pitch ees' > #<Pitch f' >)
   (#<procedure line-markup (layout props args)>
    ("m"))
```

```
 (#<procedure line-markup (layout props args)>
  ((#<procedure normal-size-super-markup (layout props arg)>
    "sus4"))))
((#<Pitch c' > #<Pitch d' > #<Pitch ees' >)
 (#<procedure line-markup (layout props args)>
  ("m"))
 (#<procedure line-markup (layout props args)>
  ((#<procedure normal-size-super-markup (layout props arg)>
    "sus2")))))
```

- Set translator property `chordNameExceptions` to:

```
'(((#<Pitch e' > #<Pitch gis' >)
   #<procedure line-markup (layout props args)>
   ("+"))
  ((#<Pitch ees' > #<Pitch ges' >)
   #<procedure line-markup (layout props args)>
   ((#<procedure super-markup (layout props arg)>
     "o")))
  ((#<Pitch ees' > #<Pitch ges' > #<Pitch bes' >)
   #<procedure line-markup (layout props args)>
   ((#<procedure normal-size-super-markup (layout props arg)>
     "ø")))
  ((#<Pitch ees' > #<Pitch ges' > #<Pitch beses' >)
   #<procedure line-markup (layout props args)>
   ((#<procedure super-markup (layout props arg)>
     "o7")))
  ((#<Pitch e' >
    #<Pitch g' >
    #<Pitch b' >
    #<Pitch fis'' >)
   #<procedure line-markup (layout props args)>
   ((#<procedure super-markup (layout props arg)>
     "lyd")))
  ((#<Pitch e' >
    #<Pitch g' >
    #<Pitch bes' >
    #<Pitch des'' >
    #<Pitch ees'' >
    #<Pitch fis'' >
    #<Pitch aes'' >)
   #<procedure line-markup (layout props args)>
   ((#<procedure super-markup (layout props arg)>
     "alt"))))
```

- Set translator property `chordNameFunction` to `ignatzek-chord-names`.
- Set translator property `chordNameLowercaseMinor` to `#f`.
- Set translator property `chordNameSeparator` to:

```
'(#<procedure hspace-markup (layout props amount)>
  0.5)
```

- Set translator property `chordNoteNamer` to `'()`.
- Set translator property `chordPrefixSpacer` to `0`.
- Set translator property `chordRootNamer` to `note-name->markup`.

- Set translator property `clefGlyph` to `"clefs.G"`.
- Set translator property `clefPosition` to `-2`.
- Set translator property `clefTranspositionFormatter` to `clef-transposition-markup`.
- Set translator property `completionFactor` to `unity-if-multimeasure`.
- Set translator property `crescendoSpanner` to `'hairpin`.
- Set translator property `cueClefTranspositionFormatter` to `clef-transposition-markup`.
- Set translator property `decrescendoSpanner` to `'hairpin`.
- Set translator property `defaultBarType` to `"|"`.
- Set translator property `doubleRepeatType` to `":..:"`.
- Set translator property `drumStyleTable` to `#<hash-table 29/61>`.
- Set translator property `endRepeatType` to `":|."`.
- Set translator property `explicitClefVisibility` to:
  `#(#t #t #t)`
- Set translator property `explicitCueClefVisibility` to:
  `#(#f #t #t)`
- Set translator property `explicitKeySignatureVisibility` to:
  `#(#t #t #t)`
- Set translator property `extendersOverRests` to `#t`.
- Set translator property `extraNatural` to `#t`.
- Set translator property `figuredBassFormatter` to `format-bass-figure`.
- Set translator property `fingeringOrientations` to:
  `'(up down)`
- Set translator property `firstClef` to `#t`.
- Set translator property `graceSettings` to:

```
'((Voice Stem direction 1)
  (Voice Slur direction -1)
  (Voice Stem font-size -3)
  (Voice Flag font-size -3)
  (Voice NoteHead font-size -3)
  (Voice TabNoteHead font-size -4)
  (Voice Dots font-size -3)
  (Voice Stem length-fraction 0.8)
  (Voice Stem no-stem-extend #t)
  (Voice Beam beam-thickness 0.384)
  (Voice Beam length-fraction 0.8)
  (Voice Accidental font-size -4)
  (Voice AccidentalCautionary font-size -4)
  (Voice Script font-size -3)
  (Voice Fingering font-size -8)
  (Voice StringNumber font-size -8))
```

- Set translator property `harmonicAccidentals` to `#t`.
- Set translator property `highStringOne` to `#t`.
- Set translator property `initialTimeSignatureVisibility` to:
  `#(#f #t #t)`
- Set translator property `instrumentTransposition` to `#<Pitch c' >`.

- Set translator property `keepAliveInterfaces` to:

```
'(bass-figure-interface
   chord-name-interface
   cluster-beacon-interface
   fret-diagram-interface
   lyric-syllable-interface
   note-head-interface
   tab-note-head-interface
   lyric-interface
   percent-repeat-item-interface
   percent-repeat-interface
   stanza-number-interface)
```

- Set translator property `keyAlterationOrder` to:

```
'((6 . -1/2)
  (2 . -1/2)
  (5 . -1/2)
  (1 . -1/2)
  (4 . -1/2)
  (0 . -1/2)
  (3 . -1/2)
  (3 . 1/2)
  (0 . 1/2)
  (4 . 1/2)
  (1 . 1/2)
  (5 . 1/2)
  (2 . 1/2)
  (6 . 1/2)
  (6 . -1)
  (2 . -1)
  (5 . -1)
  (1 . -1)
  (4 . -1)
  (0 . -1)
  (3 . -1)
  (3 . 1)
  (0 . 1)
  (4 . 1)
  (1 . 1)
  (5 . 1)
  (2 . 1)
  (6 . 1))
```

- Set translator property `lyricMelismaAlignment` to `-1`.
- Set translator property `majorSevenSymbol` to:

```
'(#<procedure line-markup (layout props args)>
  ((#<procedure triangle-markup (layout props filled)>
    #f)))
```

- Set translator property `markFormatter` to `format-mark-letters`.
- Set translator property `melismaBusyProperties` to:

```
'(melismaBusy
   slurMelismaBusy
```

```
    tieMelismaBusy
    beamMelismaBusy
    completionBusy)
```

- Set translator property `metronomeMarkFormatter` to `format-metronome-markup`.
- Set translator property `middleCClefPosition` to `-6`.
- Set translator property `middleCPosition` to `-6`.
- Set translator property `minorChordModifier` to:

```
'(#<procedure simple-markup (layout props str)>
  "m")
```

- Set translator property `noChordSymbol` to:

```
'(#<procedure simple-markup (layout props str)>
  "N.C.")
```

- Set translator property `noteToFretFunction` to `determine-frets`.
- Set translator property `partCombineTextsOnNote` to `#t`.
- Set translator property `pedalSostenutoStrings` to:

```
'("Sost. Ped." "*Sost. Ped." "*")
```

- Set translator property `pedalSostenutoStyle` to `'mixed`.
- Set translator property `pedalSustainStrings` to:

```
'("Ped." "*Ped." "*")
```

- Set translator property `pedalSustainStyle` to `'text`.
- Set translator property `pedalUnaCordaStrings` to:

```
'("una corda" "" "tre corde")
```

- Set translator property `pedalUnaCordaStyle` to `'text`.
- Set translator property `predefinedDiagramTable` to `#f`.
- Set translator property `printKeyCancellation` to `#t`.
- Set translator property `printPartCombineTexts` to `#t`.
- Set translator property `quotedCueEventTypes` to:

```
'(note-event
   rest-event
   tie-event
   beam-event
   tuplet-span-event)
```

- Set translator property `quotedEventTypes` to:

```
'(StreamEvent)
```

- Set translator property `rehearsalMark` to `1`.
- Set translator property `repeatCountVisibility` to `all-repeat-counts-visible`.
- Set translator property `scriptDefinitions` to:

```
'(("accent"
   (avoid-slur . around)
   (padding . 0.2)
   (script-stencil feta "sforzato" . "sforzato")
   (side-relative-direction . -1))
  ("accentus"
   (script-stencil feta "uaccentus" . "uaccentus")
   (side-relative-direction . -1)
```

```
 (avoid-slur . ignore)
 (padding . 0.2)
 (quantize-position . #t)
 (script-priority . -100)
 (direction . 1))
("circulus"
 (script-stencil feta "circulus" . "circulus")
 (side-relative-direction . -1)
 (avoid-slur . ignore)
 (padding . 0.2)
 (quantize-position . #t)
 (script-priority . -100)
 (direction . 1))
("coda"
 (script-stencil feta "coda" . "coda")
 (padding . 0.2)
 (avoid-slur . outside)
 (direction . 1))
("comma"
 (script-stencil feta "lcomma" . "rcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (direction . 1))
("downbow"
 (script-stencil feta "downbow" . "downbow")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (script-priority . 150))
("downmordent"
 (script-stencil
   feta
   "downmordent"
   .
   "downmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("downprall"
 (script-stencil feta "downprall" . "downprall")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("espressivo"
 (avoid-slur . around)
 (padding . 0.2)
 (script-stencil feta "espr" . "espr")
 (side-relative-direction . -1))
("fermata"
 (script-stencil feta "dfermata" . "ufermata")
```

```
 (padding . 0.2)
 (avoid-slur . around)
 (script-priority . 4000)
 (direction . 1))
("flageolet"
 (script-stencil feta "flageolet" . "flageolet")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("halfopen"
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "halfopen" . "halfopen")
 (direction . 1))
("halfopenvertical"
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil
   feta
   "halfopenvertical"
   .
   "halfopenvertical")
 (direction . 1))
("ictus"
 (script-stencil feta "ictus" . "ictus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (direction . -1))
("lheel"
 (script-stencil feta "upedalheel" . "upedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . -1))
("lineprall"
 (script-stencil feta "lineprall" . "lineprall")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("longfermata"
 (script-stencil
   feta
   "dlongfermata"
   .
   "ulongfermata")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("ltoe"
 (script-stencil feta "upedaltoe" . "upedaltoe")
```

```
 (padding . 0.2)
 (avoid-slur . around)
 (direction . -1))
("marcato"
 (script-stencil feta "dmarcato" . "umarcato")
 (padding . 0.2)
 (avoid-slur . inside)
 (quantize-position . #t)
 (side-relative-direction . -1))
("mordent"
 (script-stencil feta "mordent" . "mordent")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("open"
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "open" . "open")
 (direction . 1))
("portato"
 (script-stencil feta "uportato" . "dportato")
 (avoid-slur . around)
 (padding . 0.45)
 (side-relative-direction . -1))
("prall"
 (script-stencil feta "prall" . "prall")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("pralldown"
 (script-stencil feta "pralldown" . "pralldown")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("prallmordent"
 (script-stencil
   feta
   "prallmordent"
   .
   "prallmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("prallprall"
 (script-stencil feta "prallprall" . "prallprall")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("prallup"
 (script-stencil feta "prallup" . "prallup")
 (padding . 0.2)
 (avoid-slur . around)
```

```
 (direction . 1))
("reverseturn"
 (script-stencil
   feta
   "reverseturn"
   .
   "reverseturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (direction . 1))
("rheel"
 (script-stencil feta "dpedalheel" . "dpedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("rtoe"
 (script-stencil feta "dpedaltoe" . "dpedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("segno"
 (script-stencil feta "segno" . "segno")
 (padding . 0.2)
 (avoid-slur . outside)
 (direction . 1))
("semicirculus"
 (script-stencil
   feta
   "dsemicirculus"
   .
   "dsemicirculus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (direction . 1))
("shortfermata"
 (script-stencil
   feta
   "dshortfermata"
   .
   "ushortfermata")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("signumcongruentiae"
 (script-stencil
   feta
   "dsignumcongruentiae"
   .
   "usignumcongruentiae")
```

```
 (padding . 0.2)
 (avoid-slur . outside)
 (direction . 1))
("snappizzicato"
 (script-stencil
   feta
   "snappizzicato"
   .
   "snappizzicato")
 (padding . 0.2)
 (avoid-slur . outside)
 (direction . 1))
("staccatissimo"
 (avoid-slur . inside)
 (quantize-position . #t)
 (script-stencil
   feta
   "dstaccatissimo"
   .
   "ustaccatissimo")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.1)
 (side-relative-direction . -1)
 (toward-stem-shift . 1.0)
 (toward-stem-shift-in-column . 0.0))
("staccato"
 (script-stencil feta "staccato" . "staccato")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . inside)
 (toward-stem-shift . 1.0)
 (toward-stem-shift-in-column . 0.0)
 (padding . 0.2)
 (skyline-horizontal-padding . 0.1)
 (script-priority . -100))
("stopped"
 (script-stencil feta "stopped" . "stopped")
 (avoid-slur . inside)
 (padding . 0.2)
 (direction . 1))
("tenuto"
 (script-stencil feta "tenuto" . "tenuto")
 (quantize-position . #t)
 (avoid-slur . inside)
 (padding . 0.2)
 (side-relative-direction . -1))
("trill"
 (script-stencil feta "trill" . "trill")
 (direction . 1)
 (padding . 0.2)
 (avoid-slur . outside)
 (script-priority . 2000))
```

```
("turn"
 (script-stencil feta "turn" . "turn")
 (avoid-slur . inside)
 (padding . 0.2)
 (direction . 1))
("upbow"
 (script-stencil feta "upbow" . "upbow")
 (avoid-slur . around)
 (padding . 0.2)
 (direction . 1)
 (script-priority . 150))
("upmordent"
 (script-stencil feta "upmordent" . "upmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("upprall"
 (script-stencil feta "upprall" . "upprall")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1))
("varcoda"
 (script-stencil feta "varcoda" . "varcoda")
 (padding . 0.2)
 (avoid-slur . outside)
 (direction . 1))
("varcomma"
 (script-stencil feta "lvarcomma" . "rvarcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (direction . 1))
("verylongfermata"
 (script-stencil
   feta
   "dverylongfermata"
   .
   "uverylongfermata")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1)))
```

- Set translator property `slashChordSeparator` to:

  ```
  '(#<procedure simple-markup (layout props str)>
    "/")
  ```

- Set translator property `soloIIText` to `"Solo II"`.
- Set translator property `soloText` to `"Solo"`.
- Set translator property `startRepeatType` to `".|:"`.
- Set translator property `stringNumberOrientations` to:

  ```
  '(up down)
  ```

- Set translator property `stringOneTopmost` to `#t`.

- Set translator property `stringTunings` to:

```
'(#<Pitch e' >
  #<Pitch b >
  #<Pitch g >
  #<Pitch d >
  #<Pitch a, >
  #<Pitch e, >)
```

- Set translator property `strokeFingerOrientations` to:

```
'(right)
```

- Set translator property `subdivideBeams` to `#f`.
- Set translator property `systemStartDelimiter` to `'SystemStartBar`.
- Set translator property `tablatureFormat` to `fret-number-tablature-format`.
- Set translator property `tabStaffLineLayoutFunction` to `tablature-position-on-lines`.
- Set translator property `tieWaitForNote` to `#f`.
- Set translator property `timeSignatureFraction` to:

```
'(4 . 4)
```

- Set translator property `timeSignatureSettings` to:

```
'(((2 . 2) (beamExceptions (end (1/32 8 8 8 8))))
  ((3 . 2)
   (beamExceptions (end (1/32 8 8 8 8 8 8))))
  ((3 . 4)
   (beamExceptions (end (1/8 6) (1/12 3 3 3))))
  ((3 . 8) (beamExceptions (end (1/8 3))))
  ((4 . 2)
   (beamExceptions (end (1/16 4 4 4 4 4 4 4 4))))
  ((4 . 4)
   (beamExceptions (end (1/8 4 4) (1/12 3 3 3 3))))
  ((4 . 8) (beatStructure 2 2))
  ((6 . 4)
   (beamExceptions (end (1/16 4 4 4 4 4 4))))
  ((9 . 4)
   (beamExceptions (end (1/32 8 8 8 8 8 8 8 8))))
  ((12 . 4)
   (beamExceptions
     (end (1/32 8 8 8 8 8 8 8 8 8 8 8 8))))
  ((5 . 8) (beatStructure 3 2))
  ((8 . 8) (beatStructure 3 3 2)))
```

- Set translator property `timing` to `#t`.
- Set translator property `topLevelAlignment` to `#t`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context Score can contain Section 2.1.1 [ChoirStaff], page 57, Section 2.1.2 [ChordNames], page 58, Section 2.1.4 [Devnull], page 74, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.9 [FretBoards], page 98, Section 2.1.11 [GrandStaff], page 101, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.19 [NoteNames], page 178, Section 2.1.21 [OneStaff], page 183, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.24 [PianoStaff], page 208, Section 2.1.25

[RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.28 [StaffGroup], page 246, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

This context is built from the following engraver(s):

`Section 2.2.8 [Bar_number_engraver], page 310`
A bar number is created whenever `measurePosition` is zero and when there is a bar line (i.e., when `whichBar` is set). It is put on top of all staves, and appears only at the left side of the staff. The staves are taken from `stavesFound`, which is maintained by Section 2.2.113 [Staff_collecting_engraver], page 347.

Music types accepted:

Section 1.2.2 [alternative-event], page 41,

Properties (read)

`alternativeNumberingStyle` (symbol)
The style of an alternative's bar numbers. Can be `numbers` for going back to the same number or `numbers-with-letters` for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

`barNumberFormatter` (procedure)
A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

The following procedures are predefined:

`all-bar-numbers-visible`
Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

`first-bar-number-invisible`
Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

`first-bar-number-invisible-save-broken-bars`
Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

`first-bar-number-invisible-and-no-parenthesized-bar-`

`numbers`
: Enable bar numbers for all bars except the first bar and broken bars. This is the default.

`(every-nth-bar-number-visible n)`
: Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

`(modulo-bar-number-visible n m)`
: If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

This engraver creates the following layout object(s):

Section 3.1.12 [BarNumber], page 385.

`Section 2.2.9 [Beam_collision_engraver], page 312`
: Help beams avoid colliding with notes and clefs in other voices.

`Section 2.2.13 [Break_align_engraver], page 313`
: Align grobs with corresponding `break-align-symbols` into groups, and order the groups according to `breakAlignOrder`. The left edge of the alignment gets a separate group, with a symbol `left-edge`.

  This engraver creates the following layout object(s):

  Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, and Section 3.1.63 [LeftEdge], page 447.

`Section 2.2.22 [Concurrent_hairpin_engraver], page 317`
: Collect concurrent hairpins.

Section 2.2.26 [Default_bar_line_engraver], page 318
: This engraver determines what kind of automatic bar lines should be produced, and sets `whichBar` accordingly. It should be at the same level as Section 2.2.130 [Timing_translator], page 352.

  Properties (read)

  `automaticBars` (boolean)
  : If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

  `barAlways` (boolean)
  : If set to true a bar line is drawn after each note.

  `defaultBarType` (string)
  : Set the default type of bar line. See `whichBar` for information on available bar types.

    This variable is read by Section "Timing_translator" in *Internals Reference* at Section "Score" in *Internals Reference* level.

  `measureLength` (moment)
  : Length of one measure in the current time signature.

  `measurePosition` (moment)
  : How much of the current measure have we had. This can be set manually to create incomplete measures.

  `timing` (boolean)
  : Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

  `whichBar` (string)
  : This property is read to determine what type of bar line to create.

    Example:

    ```
    \set Staff.whichBar = ".|:"
    ```

    This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Section 2.2.42 [Footnote_engraver], page 324
: Create footnote texts.

  Properties (read)

  `currentMusicalColumn` (graphical (layout) object)
  : Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

  This engraver creates the following layout object(s):

Section 3.1.45 [FootnoteItem], page 426, and Section 3.1.46 [FootnoteSpanner], page 427.

`Section 2.2.49 [Grace_spacing_engraver], page 327`
Bookkeeping of shortest starting and playing notes in grace note runs.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.49 [GraceSpacing], page 431.

`Section 2.2.66 [Mark_engraver], page 332`
Create `RehearsalMark` objects. It puts them on top of all staves (which is taken from the property `stavesFound`). If moving this engraver to a different context, Section 2.2.113 [Staff_collecting_engraver], page 347, must move along, otherwise all marks end up on the same Y location.

Music types accepted:

Section 1.2.35 [mark-event], page 45,

Properties (read)

`markFormatter` (procedure)
A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`rehearsalMark` (integer)
The last rehearsal mark printed.

`stavesFound` (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s):

Section 3.1.90 [RehearsalMark], page 477.

`Section 2.2.72 [Metronome_mark_engraver], page 333`
Engrave metronome marking. This delegates the formatting work to the function in the `metronomeMarkFormatter` property. The mark is put over all staves. The staves are taken from the `stavesFound` property, which is maintained by Section 2.2.113 [Staff_collecting_engraver], page 347.

Music types accepted:

Section 1.2.68 [tempo-change-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

This engraver creates the following layout object(s):

Section 3.1.73 [MetronomeMark], page 458.

`Section 2.2.82 [Output_property_engraver], page 338`

Apply a procedure to any grob acknowledged.

Music types accepted:

Section 1.2.4 [apply-output-event], page 42,

`Section 2.2.84 [Paper_column_engraver], page 338`

Take care of generating columns.

This engraver decides whether a column is breakable. The default is that a column is always breakable. However, every `Bar_engraver` that does not have a barline at a certain point will set `forbidBreaks` in the score context to stop line breaks. In practice, this means that you can make a break point by creating a bar line (assuming that there are no beams or notes that prevent a break point).

Music types accepted:

Section 1.2.12 [break-event], page 42, and Section 1.2.29 [label-event], page 44,

Properties (read)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

Properties (write)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.77 [NonMusicalPaperColumn], page 464, and Section 3.1.84 [PaperColumn], page 470.

`Section 2.2.85 [Parenthesis_engraver], page 339`

Parenthesize objects whose music cause has the `parenthesize` property.

This engraver creates the following layout object(s):

Section 3.1.85 [ParenthesesItem], page 471.

Section 2.2.95 [Repeat_acknowledge_engraver], page 342

Acknowledge repeated music, and convert the contents of `repeatCommands` into an appropriate setting for `whichBar`.

Properties (read)

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta` *x*`)`, where *x* is a string or `#f`. `'end-repeat` is also accepted as a command.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is '`S.|:`'.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Section 2.2.107 [Spacing_engraver], page 346

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted:

Section 1.2.60 [spacing-section-event], page 48,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s):

Section 3.1.102 [SpacingSpanner], page 489.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
: Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`Section 2.2.116 [Stanza_number_align_engraver], page 347`
: This engraver ensures that stanza numbers are neatly aligned.

`Section 2.2.119 [System_start_delimiter_engraver], page 348`
: Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`Section 2.2.130 [Timing_translator], page 352`
: This engraver adds the alias `Timing` to its containing context. Responsible for synchronizing timing information from staves. Normally in `Score`. In order to create polyrhythmic music, this engraver should be removed from `Score` and placed in `Staff`.

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

Properties (write)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`Section 2.2.133 [Tweak_engraver], page 354`
Read the `tweaks` property from the originating event, and set properties.

`Section 2.2.135 [Vertical_align_engraver], page 354`
Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

`Section 2.2.136 [Volta_engraver], page 355`
Make volta brackets.

Properties (read)

`repeatCommands` (list)
This property is a list of commands of the form `(list 'volta x)`, where *x* is a string or `#f`. `'end-repeat` is also accepted as a command.

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`voltaSpannerDuration` (moment)
This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

This engraver creates the following layout object(s):

Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

2.1.27 Staff

Handles clefs, bar lines, keys, accidentals. It can contain `Voice` contexts.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner],

page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set translator property `createSpacing` to `#t`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.33 [Voice], page 293.

Context Staff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.20 [NullVoice], page 180, and Section 2.1.33 [Voice], page 293.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

Properties (read)

`accidentalGrouping` (symbol)

If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`autoAccidentals` (list)

List of different ways to typeset an accidental.

For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

Each entry in the list is either a symbol or a procedure.

symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave . name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber . measureposition`*`))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`
Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.27 [Dot_column_engraver], page 318`
: Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
: Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`

Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`* `.` *`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`* `.` *`alter`*`)` or `((`*`octave`* `.` *`step`*`)` `.` *`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
: Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
: Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
See `pedalSustainStyle`.

`pedalSustainStrings` (list)
A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
: break visibility for the initial time signature.

`partialBusy` (boolean)
: Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.28 StaffGroup

Groups staves while adding a bracket on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically. `StaffGroup` only consists of a collection of staves, with a bracket in front and spanning bar lines.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.55 [InstrumentName], page 436, Section 3.1.103 [SpanBar], page 490, Section 3.1.104 [SpanBarStub], page 491, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, and Section 3.1.136 [VerticalAlignment], page 528.

This context sets the following properties:

- Set grob-property `extra-spacing-width` in Section 3.1.39 [DynamicText], page 418, to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `shortInstrumentName` to `'()`.
- Set translator property `systemStartDelimiter` to `'SystemStartBracket`.
- Set translator property `topLevelAlignment` to `#f`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.27 [Staff], page 235.

Context StaffGroup can contain Section 2.1.1 [ChoirStaff], page 57, Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.8 [FiguredBass], page 96, Section 2.1.9 [FretBoards], page 98, Section 2.1.11 [GrandStaff], page 101, Section 2.1.16 [Lyrics], page 151, Section 2.1.21 [OneStaff], page 183, Section 2.1.24 [PianoStaff], page 208, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.28 [StaffGroup], page 246, and Section 2.1.29 [TabStaff], page 248.

This context is built from the following engraver(s):

`Section 2.2.55 [Instrument_name_engraver], page 328`
: Create a system start text for instrument or vocal names.

  Properties (read)

  `currentCommandColumn` (graphical (layout) object)
  : Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.108 [Span_arpeggio_engraver], page 346`
: Make arpeggios that span multiple staves.

Properties (read)

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.109 [Span_bar_engraver], page 346`
: Make cross-staff bar lines: It catches all normal bar lines and draws a single span bar across them.

This engraver creates the following layout object(s):

Section 3.1.103 [SpanBar], page 490.

`Section 2.2.110 [Span_bar_stub_engraver], page 346`
: Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s):

Section 3.1.104 [SpanBarStub], page 491.

`Section 2.2.119 [System_start_delimiter_engraver], page 348`
: Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`Section 2.2.135 [Vertical_align_engraver], page 354`

Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

2.1.29 TabStaff

Context for generating tablature. It accepts only `TabVoice` contexts and handles the line spacing, the tablature clef etc. properly.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.126 [TimeSignature], page 515, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `after-line-breaking` in Section 3.1.92 [RepeatTie], page 480, to `repeat-tie::handle-tab-note-head`.
- Set grob-property `after-line-breaking` in Section 3.1.124 [Tie], page 513, to `tie::handle-tab-note-head`.
- Set grob-property `avoid-note-head` in Section 3.1.109 [Stem], page 494, to `#t`.
- Set grob-property `beam-thickness` in Section 3.1.19 [Beam], page 390, to `0.32`.

- Set grob-property `beam-thickness` in Section 3.1.111 [StemTremolo], page 497, to `0.32`.
- Set grob-property `beam-width` in Section 3.1.111 [StemTremolo], page 497, to `stem-tremolo::calc-tab-width`.
- Set grob-property `bound-details.left` in Section 3.1.48 [Glissando], page 430, to:
  ```
  '((attach-dir . 1) (padding . 0.3))
  ```
- Set grob-property `bound-details.right` in Section 3.1.48 [Glissando], page 430, to:
  ```
  '((attach-dir . -1) (padding . 0.3))
  ```
- Set grob-property `details` in Section 3.1.109 [Stem], page 494, to:
  ```
  '((lengths 0 0 0 0 0 0)
    (beamed-lengths 0 0 0)
    (beamed-minimum-free-lengths 0 0 0)
    (beamed-extreme-minimum-free-lengths 0 0)
    (stem-shorten 0 0))
  ```
- Set grob-property `extra-dy` in Section 3.1.48 [Glissando], page 430, to `glissando::calc-tab-extra-dy`.
- Set grob-property `glyph-name` in Section 3.1.121 [TabNoteHead], page 508, to `tab-note-head::calc-glyph-name`.
- Set grob-property `ignore-collision` in Section 3.1.79 [NoteColumn], page 466, to `#t`.
- Set grob-property `length-fraction` in Section 3.1.19 [Beam], page 390, to `0.62`.
- Set grob-property `length-fraction` in Section 3.1.111 [StemTremolo], page 497, to `#<procedure #f (grob)>`.
- Set grob-property `no-stem-extend` in Section 3.1.109 [Stem], page 494, to `#t`.
- Set grob-property `staff-space` in Section 3.1.107 [StaffSymbol], page 493, to `1.5`.
- Set grob-property `stencil` in Section 3.1.9 [Arpeggio], page 379, to `#f`.
- Set grob-property `stencil` in Section 3.1.19 [Beam], page 390, to `#f`.
- Set grob-property `stencil` in Section 3.1.25 [Clef], page 397, to `clef::print-modern-tab-if-set`.
- Set grob-property `stencil` in Section 3.1.34 [Dots], page 412, to `#f`.
- Set grob-property `stencil` in Section 3.1.40 [DynamicTextSpanner], page 420, to `#f`.
- Set grob-property `stencil` in Section 3.1.39 [DynamicText], page 418, to `#f`.
- Set grob-property `stencil` in Section 3.1.44 [Flag], page 425, to `#f`.
- Set grob-property `stencil` in Section 3.1.48 [Glissando], page 430, to `glissando::draw-tab-glissando`.
- Set grob-property `stencil` in Section 3.1.52 [Hairpin], page 432, to `#f`.
- Set grob-property `stencil` in Section 3.1.60 [LaissezVibrerTie], page 445, to `#f`.
- Set grob-property `stencil` in Section 3.1.75 [MultiMeasureRestNumber], page 461, to `#f`.
- Set grob-property `stencil` in Section 3.1.76 [MultiMeasureRestText], page 463, to `#f`.
- Set grob-property `stencil` in Section 3.1.74 [MultiMeasureRest], page 459, to `#f`.
- Set grob-property `stencil` in Section 3.1.88 [PhrasingSlur], page 474, to `#f`.
- Set grob-property `stencil` in Section 3.1.92 [RepeatTie], page 480, to `#f`.
- Set grob-property `stencil` in Section 3.1.94 [Rest], page 481, to `#f`.
- Set grob-property `stencil` in Section 3.1.96 [Script], page 483, to `#f`.
- Set grob-property `stencil` in Section 3.1.99 [Slur], page 485, to `slur::draw-tab-slur`.
- Set grob-property `stencil` in Section 3.1.111 [StemTremolo], page 497, to `#f`.

- Set grob-property `stencil` in Section 3.1.109 [Stem], page 494, to `#f`.
- Set grob-property `stencil` in Section 3.1.121 [TabNoteHead], page 508, to `tab-note-head::whiteout-if-style-set`.
- Set grob-property `stencil` in Section 3.1.122 [TextScript], page 510, to `#f`.
- Set grob-property `stencil` in Section 3.1.123 [TextSpanner], page 512, to `#f`.
- Set grob-property `stencil` in Section 3.1.124 [Tie], page 513, to `#f`.
- Set grob-property `stencil` in Section 3.1.126 [TimeSignature], page 515, to `#f`.
- Set grob-property `stencil` in Section 3.1.131 [TupletBracket], page 522, to `#f`.
- Set grob-property `stencil` in Section 3.1.132 [TupletNumber], page 524, to `#f`.
- Set grob-property `style` in Section 3.1.44 [Flag], page 425, to `'no-flag`.
- Set translator property `autoBeaming` to `#f`.
- Set translator property `clefGlyph` to `"clefs.tab"`.
- Set translator property `clefPosition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `handleNegativeFrets` to `'recalculate`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `restrainOpenStrings` to `#f`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.30 [TabVoice], page 257.

Context TabStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.20 [NullVoice], page 180, and Section 2.1.30 [TabVoice], page 257.

This context is built from the following engraver(s):

`Section 2.2.5 [Axis_group_engraver], page 309`
: Group all objects created in this context in a `VerticalAxisGroup` spanner.

  Properties (read)

  `currentCommandColumn` (graphical (layout) object)
  : Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

  `hasAxisGroup` (boolean)
  : True if the current context is contained in an axis group.

  `keepAliveInterfaces` (list)
  : A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

  Properties (write)

  `hasAxisGroup` (boolean)
  : True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`
Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`
Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
Name of the symbol within the music font.

`cueClefPosition` (number)
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
'`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.27 [Dot_column_engraver], page 318`
Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.62 [Ledger_line_engraver], page 331`
Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
: Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
: Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(end-moment . grob)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
: Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.121 [Tab_staff_symbol_engraver], page 350`
Create a tablature staff symbol, but look at `stringTunings` for the number of lines.

Properties (read)

`stringTunings` (list)
The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Section 2.2.128 [Time_signature_engraver], page 352`
Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
break visibility for the initial time signature.

`partialBusy` (boolean)
: Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

2.1.30 TabVoice

Context for drawing notes in a Tab staff.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.121 [TabNoteHead], page 508, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`
: Generate an Arpeggio symbol.

  Music types accepted:

  Section 1.2.5 [arpeggio-event], page 42,

  This engraver creates the following layout object(s):

  Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`
Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
If set to true then beams are generated automatically.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
Signal if a beam is present.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.41 [Font_size_engraver], page 323`

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`

Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`
Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.63 [Ligature_bracket_engraver], page 331`
Handle `Ligature_events` by engraving `Ligature` brackets.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.64 [LigatureBracket], page 449.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.76 [Note_head_line_engraver], page 336`
: Engrave a line between two note heads in a staff switch if `followVoice` is set.

  Properties (read)

  `followVoice` (boolean)
  : If set, note heads are tracked across staff switches by a thin line.

  This engraver creates the following layout object(s):

  Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.80 [Note_spacing_engraver], page 337`
: Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

  This engraver creates the following layout object(s):

  Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
: Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

  Music types accepted:

  Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

  Properties (read)

  `aDueText` (markup)
  : Text to print at a unisono passage.

  `partCombineTextsOnNote` (boolean)
  : Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`

Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`

Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.96 [Repeat_tie_engraver], page 342`

Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
: Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
: Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
: Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.120 [Tab_note_heads_engraver], page 349`
: Generate one or more tablature note heads from event of type `NoteEvent`.

Music types accepted:

Section 1.2.23 [fingering-event], page 44, Section 1.2.41 [note-event], page 46, and Section 1.2.65 [string-number-event], page 49,

Properties (read)

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

This engraver creates the following layout object(s):

Section 3.1.121 [TabNoteHead], page 508.

`Section 2.2.122 [Tab_tie_follow_engraver], page 350`
: Adjust TabNoteHead properties when a tie is followed by a slur or glissando.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`
Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`
Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.1.31 VaticanaStaff

Same as `Staff` context, except that it is accommodated for typesetting Gregorian Chant in the notational style of Editio Vaticana.

This context also accepts commands for the following context(s):

Staff.

This context creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.11 [BarLine], page 381, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.33 [DotColumn], page 412, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.55 [InstrumentName], page 436, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.78 [NoteCollision], page 465, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.95 [RestCollision], page 483, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.137 [VerticalAxisGroup], page 528.

This context sets the following properties:

- Set grob-property `glyph-name-alist` in Section 3.1.1 [Accidental], page 370, to:
  ```
  '((-1/2 . "accidentals.vaticanaM1")
    (0 . "accidentals.vaticana0")
    (1/2 . "accidentals.mensural1"))
  ```
- Set grob-property `glyph-name-alist` in Section 3.1.58 [KeySignature], page 441, to:
  ```
  '((-1/2 . "accidentals.vaticanaM1")
    (0 . "accidentals.vaticana0")
    (1/2 . "accidentals.mensural1"))
  ```
- Set grob-property `line-count` in Section 3.1.107 [StaffSymbol], page 493, to `4`.
- Set grob-property `neutral-direction` in Section 3.1.32 [Custos], page 410, to `-1`.
- Set grob-property `neutral-position` in Section 3.1.32 [Custos], page 410, to `3`.
- Set grob-property `style` in Section 3.1.32 [Custos], page 410, to `'vaticana`.
- Set grob-property `style` in Section 3.1.34 [Dots], page 412, to `'vaticana`.
- Set grob-property `thickness` in Section 3.1.107 [StaffSymbol], page 493, to `0.6`.
- Set grob-property `transparent` in Section 3.1.11 [BarLine], page 381, to `#t`.
- Set translator property `clefGlyph` to `"clefs.vaticana.do"`.
- Set translator property `clefPosition` to `1`.
- Set translator property `clefTransposition` to `0`.
- Set translator property `createSpacing` to `#t`.
- Set translator property `ignoreFiguredBassRest` to `#f`.
- Set translator property `instrumentName` to `'()`.
- Set translator property `localAlterations` to `'()`.
- Set translator property `middleCClefPosition` to `1`.
- Set translator property `middleCPosition` to `1`.
- Set translator property `shortInstrumentName` to `'()`.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Section 2.1.32 [VaticanaVoice], page 281.

Context VaticanaStaff can contain Section 2.1.3 [CueVoice], page 60, Section 2.1.20 [NullVoice], page 180, and Section 2.1.32 [VaticanaVoice], page 281.

This context is built from the following engraver(s):

`Section 2.2.1 [Accidental_engraver], page 306`

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

Properties (read)

`accidentalGrouping` (symbol)

If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`autoAccidentals` (list)

List of different ways to typeset an accidental.

For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

Each entry in the list is either a symbol or a procedure.

symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
: The current key signature. This is an alist containing `(step . alter)` or `((octave . step) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((octave . name) . (alter barnumber . measureposition))` pairs.

Properties (write)

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((octave . name) . (alter barnumber . measureposition))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Section 2.2.5 [Axis_group_engraver], page 309`
Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Section 2.2.7 [Bar_engraver], page 310`
Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point.

This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Section 2.2.17 [Clef_engraver], page 314`

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Section 2.2.19 [Collision_engraver], page 315`

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Section 2.2.24 [Cue_clef_engraver], page 317`
Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
Name of the symbol within the music font.

`cueClefPosition` (number)
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
'`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Section 2.2.25 [Custos_engraver], page 317`
Engrave custodes.

This engraver creates the following layout object(s):

Section 3.1.32 [Custos], page 410.

`Section 2.2.27 [Dot_column_engraver], page 318`
Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Section 2.2.37 [Figured_bass_engraver], page 322`
Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Section 2.2.38 [Figured_bass_position_engraver], page 323`
Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Section 2.2.39 [Fingering_column_engraver], page 323`
Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.55 [Instrument_name_engraver], page 328`
Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Section 2.2.58 [Key_engraver], page 329`
Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step`*` . `*`alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Section 2.2.62 [Ledger_line_engraver], page 331`
: Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Section 2.2.81 [Ottava_spanner_engraver], page 337`
: Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Section 2.2.89 [Piano_pedal_align_engraver], page 340`
Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Section 2.2.90 [Piano_pedal_engraver], page 340`
Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Section 2.2.94 [Pure_from_neighbor_engraver], page 342`
: Coordinates items that get their pure heights from their neighbors.

`Section 2.2.97 [Rest_collision_engraver], page 343`
: Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Section 2.2.102 [Script_row_engraver], page 344`
: Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Section 2.2.103 [Separating_line_group_engraver], page 344`
: Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Section 2.2.113 [Staff_collecting_engraver], page 347`
: Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
A list of all staff-symbols found.

`Section 2.2.115 [Staff_symbol_engraver], page 347`
Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

2.1.32 VaticanaVoice

Same as `Voice` context, except that it is accommodated for typesetting Gregorian Chant in the notational style of Editio Vaticana.

This context also accepts commands for the following context(s):

Voice.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.33 [DotColumn], page 412, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.42 [Fingering], page 422, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, Section 3.1.135 [VaticanaLigature], page 527, and Section 3.1.138 [VoiceFollower], page 530.

This context sets the following properties:

- Set grob-property `padding` in Section 3.1.96 [Script], page 483, to `0.5`.
- Set grob-property `style` in Section 3.1.80 [NoteHead], page 467, to `'vaticana.punctum`.
- Set translator property `autoBeaming` to `#f`.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`
Generate an Arpeggio symbol.

Music types accepted:

Section 1.2.5 [arpeggio-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`
Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
If set to true then beams are generated automatically.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
Signal if a beam is present.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
If set, produce counters for percent repeats.

`measureLength` (moment)
Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.35 [Episema_engraver], page 321`
Create an *Editio Vaticana*-style episema line.

Music types accepted:

Section 1.2.21 [episema-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.41 [Episema], page 422.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*`end-moment . grob`*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`
Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`
Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`
: Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

  Properties (read)

  `fingeringOrientations` (list)
  : A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

  `harmonicDots` (boolean)
  : If set, harmonic notes in dotted chords get dots.

  `stringNumberOrientations` (list)
  : See `fingeringOrientations`.

  `strokeFingerOrientations` (list)
  : See `fingeringOrientations`.

  This engraver creates the following layout object(s):

  Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`
Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`
Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.124 [Text_engraver], page 350`
Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.126 [Tie_engraver], page 351`
Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`
Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`
Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

`Section 2.2.134 [Vaticana_ligature_engraver], page 354`
: Handle ligatures by glueing special ligature heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45, and Section 1.2.48 [pes-or-flexa-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412, and Section 3.1.135 [VaticanaLigature], page 527.

2.1.33 Voice

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and subscripts, slurs, ties, and rests.

You have to instantiate this explicitly if you want to have multiple voices on the same staff.

This context creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.23 [BreathingSign], page 394, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124

[Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, and Section 3.1.138 [VoiceFollower], page 530.

This is a 'Bottom' context; no contexts will be created implicitly from it.

This context cannot contain other contexts.

This context is built from the following engraver(s):

`Section 2.2.3 [Arpeggio_engraver], page 308`
: Generate an Arpeggio symbol.

  Music types accepted:

  Section 1.2.5 [arpeggio-event], page 42,

  This engraver creates the following layout object(s):

  Section 3.1.9 [Arpeggio], page 379.

`Section 2.2.4 [Auto_beam_engraver], page 308`
: Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

  Music types accepted:

  Section 1.2.9 [beam-forbid-event], page 42,

  Properties (read)

  `autoBeaming` (boolean)
  : If set to true then beams are generated automatically.

  `baseMoment` (moment)
  : Smallest unit of time that will stand on its own as a subdivided section.

  `beamExceptions` (list)
  : An alist of exceptions to autobeam rules that normally end on beats.

  `beamHalfMeasure` (boolean)
  : Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

  `beatStructure` (list)
  : List of `baseMoment`s that are combined to make beats.

  `subdivideBeams` (boolean)
  : If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

  This engraver creates the following layout object(s):

  Section 3.1.19 [Beam], page 390.

`Section 2.2.10 [Beam_engraver], page 312`
Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
Signal if a beam is present.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.12 [Bend_engraver], page 312`
Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Section 2.2.14 [Breathing_sign_engraver], page 313`
Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Section 2.2.16 [Chord_tremolo_engraver], page 314`
Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.18 [Cluster_spanner_engraver], page 315`
Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Section 2.2.28 [Dots_engraver], page 319`
Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Section 2.2.29 [Double_percent_repeat_engraver], page 319`
Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Section 2.2.32 [Dynamic_align_engraver], page 320`
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Section 2.2.33 [Dynamic_engraver], page 320`
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Section 2.2.40 [Fingering_engraver], page 323`
Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Section 2.2.41 [Font_size_engraver], page 323`
Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Section 2.2.43 [Forbid_line_break_engraver], page 324`
Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Section 2.2.45 [Glissando_engraver], page 325`

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of noteheads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Section 2.2.46 [Grace_auto_beam_engraver], page 325`

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.47 [Grace_beam_engraver], page 326`

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoments` that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Section 2.2.48 [Grace_engraver], page 326`
Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Section 2.2.52 [Grob_pq_engraver], page 327`
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Section 2.2.56 [Instrument_switch_engraver], page 329`
Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Section 2.2.61 [Laissez_vibrer_engraver], page 330`
Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Section 2.2.63 [Ligature_bracket_engraver], page 331`
Handle `Ligature_events` by engraving `Ligature` brackets.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.64 [LigatureBracket], page 449.

`Section 2.2.74 [Multi_measure_rest_engraver], page 335`

Engrave multi-measure rests that are produced with '`R`'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Section 2.2.75 [New_fingering_engraver], page 335`

Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

Properties (read)

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where

fingerings are put relative to the chord being fingered.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`Section 2.2.76 [Note_head_line_engraver], page 336`
: Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Section 2.2.77 [Note_heads_engraver], page 336`
: Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Section 2.2.80 [Note_spacing_engraver], page 337`
: Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Section 2.2.86 [Part_combine_engraver], page 339`
: Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Section 2.2.87 [Percent_repeat_engraver], page 339`

Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Section 2.2.88 [Phrasing_slur_engraver], page 340`

Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Section 2.2.93 [Pitched_trill_engraver], page 341`
Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Section 2.2.96 [Repeat_tie_engraver], page 342`
Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Section 2.2.98 [Rest_engraver], page 343`
Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Section 2.2.99 [Rhythmic_column_engraver], page 343`
Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Section 2.2.100 [Script_column_engraver], page 344`
Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Section 2.2.101 [Script_engraver], page 344`
Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Section 2.2.104 [Slash_repeat_engraver], page 345`
Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Section 2.2.105 [Slur_engraver], page 345`
Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Section 2.2.112 [Spanner_break_forbid_engraver], page 347`
Forbid breaks in certain spanners.

`Section 2.2.118 [Stem_engraver], page 348`
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
See `stemLeftBeamCount`.

`whichBar` (string)
This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Section 2.2.124 [Text_engraver], page 350`

Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Section 2.2.125 [Text_spanner_engraver], page 351`

Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Section 2.2.126 [Tie_engraver], page 351`

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Section 2.2.131 [Trill_spanner_engraver], page 353`

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Section 2.2.132 [Tuplet_engraver], page 353`

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

2.2 Engravers and Performers

See Section "Modifying context plug-ins" in *Notation Reference*.

2.2.1 Accidental_engraver

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidental at `Voice` level, so you can `\override` them at `Voice`.

Properties (read)

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is

Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoCautionaries` (list)
List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`extraNatural` (boolean)
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`harmonicAccidentals` (boolean)
If set, harmonic notes in chords get accidentals.

`internalBarNumber` (integer)
Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keyAlterations` (list)
The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`localAlterations` (list)
The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber`*` . `*`measureposition`*`))` pairs.

Properties (write)

`localAlterations` (list)
The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain `((`*`octave`*` . `*`name`*`) . (`*`alter barnumber`*` . `*`measureposition`*`))` pairs.

This engraver creates the following layout object(s):

Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, and Section 3.1.4 [AccidentalSuggestion], page 373.

`Accidental_engraver` is part of the following context(s): Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.2 Ambitus_engraver

Create an ambitus.

Properties (read)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step`*` . `*`alter`*`)` or `((`*`octave`*` . `*`step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

This engraver creates the following layout object(s):

Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.5 [Ambitus], page 375, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.7 [AmbitusLine], page 377, and Section 3.1.8 [AmbitusNoteHead], page 378.

`Ambitus_engraver` is not part of any context.

2.2.3 Arpeggio_engraver

Generate an Arpeggio symbol.

Music types accepted:

Section 1.2.5 [arpeggio-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Arpeggio_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.4 Auto_beam_engraver

Generate beams based on measure characteristics and observed Stems. Uses `baseMoment`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Section 2.2.118 [Stem_engraver], page 348, properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Auto_beam_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.5 Axis_group_engraver

Group all objects created in this context in a `VerticalAxisGroup` spanner.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

Properties (write)

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.137 [VerticalAxisGroup], page 528.

`Axis_group_engraver` is part of the following context(s): Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.8 [FiguredBass], page 96, Section 2.1.9 [FretBoards], page 98, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.19 [NoteNames], page 178, Section 2.1.21 [OneStaff], page 183, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.6 Balloon_engraver

Create balloon texts.

Music types accepted:

Section 1.2.3 [annotate-output-event], page 41,

This engraver creates the following layout object(s):

Section 3.1.10 [BalloonTextItem], page 381.

`Balloon_engraver` is not part of any context.

2.2.7 Bar_engraver

Create barlines. This engraver is controlled through the `whichBar` property. If it has no bar line to create, it will forbid a linebreak at this point. This engraver is required to trigger the creation of clefs at the start of systems.

Properties (read)

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.11 [BarLine], page 381.

`Bar_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.7 [Dynamics], page 92, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.8 Bar_number_engraver

A bar number is created whenever `measurePosition` is zero and when there is a bar line (i.e., when `whichBar` is set). It is put on top of all staves, and appears only at the left side of the staff. The staves are taken from `stavesFound`, which is maintained by Section 2.2.113 [Staff_collecting_engraver], page 347.

Music types accepted:

Section 1.2.2 [alternative-event], page 41,

Properties (read)

`alternativeNumberingStyle` (symbol)
: The style of an alternative's bar numbers. Can be `numbers` for going back to the same number or `numbers-with-letters` for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

`barNumberFormatter` (procedure)
: A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
: A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

  The following procedures are predefined:

  `all-bar-numbers-visible`
  : Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

  `first-bar-number-invisible`
  : Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

  `first-bar-number-invisible-save-broken-bars`
  : Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

  `first-bar-number-invisible-and-no-parenthesized-bar-numbers`
  : Enable bar numbers for all bars except the first bar and broken bars. This is the default.

  `(every-nth-bar-number-visible` *n*`)`
  : Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

  `(modulo-bar-number-visible` *n m*`)`
  : If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

Properties (write)

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

This engraver creates the following layout object(s):

Section 3.1.12 [BarNumber], page 385.

`Bar_number_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.9 Beam_collision_engraver

Help beams avoid colliding with notes and clefs in other voices.

`Beam_collision_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.10 Beam_engraver

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Beam_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.20 [NullVoice], page 180, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.11 Beam_performer

Music types accepted:

Section 1.2.8 [beam-event], page 42,

`Beam_performer` is not part of any context.

2.2.12 Bend_engraver

Create fall spanners.

Music types accepted:

Section 1.2.10 [bend-after-event], page 42,

This engraver creates the following layout object(s):

Section 3.1.20 [BendAfter], page 392.

`Bend_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.13 Break_align_engraver

Align grobs with corresponding `break-align-symbols` into groups, and order the groups according to `breakAlignOrder`. The left edge of the alignment gets a separate group, with a symbol `left-edge`.

This engraver creates the following layout object(s):

Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, and Section 3.1.63 [LeftEdge], page 447.

`Break_align_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.14 Breathing_sign_engraver

Create a breathing sign.

Music types accepted:

Section 1.2.14 [breathing-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.23 [BreathingSign], page 394.

`Breathing_sign_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.15 Chord_name_engraver

Catch note and rest events and generate the appropriate chordname.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`lastChord` (markup)
: Last chord, used for detecting chord changes.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

Properties (write)

`lastChord` (markup)
: Last chord, used for detecting chord changes.

This engraver creates the following layout object(s):

Section 3.1.24 [ChordName], page 396.

`Chord_name_engraver` is part of the following context(s): Section 2.1.2 [ChordNames], page 58.

2.2.16 Chord_tremolo_engraver

Generate beams for tremolo repeats.

Music types accepted:

Section 1.2.74 [tremolo-span-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Chord_tremolo_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.17 Clef_engraver

Determine and set reference point for pitches.

Properties (read)

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

This engraver creates the following layout object(s):

Section 3.1.25 [Clef], page 397, and Section 3.1.26 [ClefModifier], page 400.

`Clef_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.18 Cluster_spanner_engraver

Engrave a cluster using `Spanner` notation.

Music types accepted:

Section 1.2.15 [cluster-note-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.27 [ClusterSpanner], page 402, and Section 3.1.28 [ClusterSpannerBeacon], page 402.

`Cluster_spanner_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.19 Collision_engraver

Collect `NoteColumns`, and as soon as there are two or more, put them in a `NoteCollision` object.

This engraver creates the following layout object(s):

Section 3.1.78 [NoteCollision], page 465.

`Collision_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.20 Completion_heads_engraver

This engraver replaces `Note_heads_engraver`. It plays some trickery to break long notes and automatically tie them into the next measure.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`completionFactor` (an exact rational or procedure)
: When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.

  If `#f`, the completion engraver uses the scale-factor of each duration being split.

  If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

`completionUnit` (moment)
: Sub-bar unit of completion.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

Properties (write)

`completionBusy` (boolean)
: Whether a completion-note head is playing.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467, Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Completion_heads_engraver` is not part of any context.

2.2.21 Completion_rest_engraver

This engraver replaces `Rest_engraver`. It plays some trickery to break long rests into the next measure.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`completionFactor` (an exact rational or procedure)
: When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.

  If `#f`, the completion engraver uses the scale-factor of each duration being split.

  If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

`completionUnit` (moment)
: Sub-bar unit of completion.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

Properties (write)

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Completion_rest_engraver` is not part of any context.

2.2.22 Concurrent_hairpin_engraver

Collect concurrent hairpins.

`Concurrent_hairpin_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.23 Control_track_performer

`Control_track_performer` is not part of any context.

2.2.24 Cue_clef_engraver

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

This engraver creates the following layout object(s):

Section 3.1.26 [ClefModifier], page 400, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

`Cue_clef_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.25 Custos_engraver

Engrave custodes.

This engraver creates the following layout object(s):

Section 3.1.32 [Custos], page 410.

`Custos_engraver` is part of the following context(s): Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.26 Default_bar_line_engraver

This engraver determines what kind of automatic bar lines should be produced, and sets `whichBar` accordingly. It should be at the same level as Section 2.2.130 [Timing_translator], page 352.

Properties (read)

`automaticBars` (boolean)
: If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
: If set to true a bar line is drawn after each note.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Section "Timing_translator" in *Internals Reference* at Section "Score" in *Internals Reference* level.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

`Default_bar_line_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.27 Dot_column_engraver

Engrave dots on dotted notes shifted to the right of the note. If omitted, then dots appear on top of the notes.

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412.

`Dot_column_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.28 Dots_engraver

Create Section 3.1.34 [Dots], page 412, objects for Section 3.2.97 [rhythmic-head-interface], page 588s.

This engraver creates the following layout object(s):

Section 3.1.34 [Dots], page 412.

`Dots_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.29 Double_percent_repeat_engraver

Make double measure repeats.

Music types accepted:

Section 1.2.19 [double-percent-event], page 43,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`measureLength` (moment)
: Length of one measure in the current time signature.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.35 [DoublePercentRepeat], page 413, and Section 3.1.36 [DoublePercentRepeatCounter], page 414.

`Double_percent_repeat_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.30 Drum_note_performer

Play drum notes.

Music types accepted:

Section 1.2.41 [note-event], page 46,

`Drum_note_performer` is not part of any context.

2.2.31 Drum_notes_engraver

Generate drum note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`agostini-drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

  The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list `(`*`notehead-style script vertical-position`*`)` as values.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467, and Section 3.1.96 [Script], page 483.

`Drum_notes_engraver` is part of the following context(s): Section 2.1.6 [DrumVoice], page 81.

2.2.32 Dynamic_align_engraver

Align hairpins and dynamic texts on a horizontal line.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.38 [DynamicLineSpanner], page 417.

`Dynamic_align_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.33 Dynamic_engraver

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.13 [break-span-event], page 43, and Section 1.2.61 [span-dynamic-event], page 48,

Properties (read)

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

This engraver creates the following layout object(s):

Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

`Dynamic_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.34 Dynamic_performer

Music types accepted:

Section 1.2.1 [absolute-dynamic-event], page 41, Section 1.2.17 [crescendo-event], page 43, and Section 1.2.18 [decrescendo-event], page 43,

Properties (read)

`dynamicAbsoluteVolumeFunction` (procedure)
: A procedure that takes one argument, the text value of a dynamic event, and returns the absolute volume of that dynamic event.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(`*`min . max`*`)` pair of numbers for the loudness range of the instrument.

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`Dynamic_performer` is not part of any context.

2.2.35 Episema_engraver

Create an *Editio Vaticana*-style episema line.

Music types accepted:

Section 1.2.21 [episema-event], page 43,

This engraver creates the following layout object(s):

Section 3.1.41 [Episema], page 422.

`Episema_engraver` is part of the following context(s): Section 2.1.13 [GregorianTranscriptionVoice], page 114, and Section 2.1.32 [VaticanaVoice], page 281.

2.2.36 Extender_engraver

Create lyric extenders.

Music types accepted:

Section 1.2.16 [completize-extender-event], page 43, and Section 1.2.22 [extender-event], page 43,

Properties (read)

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s):

Section 3.1.65 [LyricExtender], page 450.

`Extender_engraver` is part of the following context(s): Section 2.1.16 [Lyrics], page 151.

2.2.37 Figured_bass_engraver

Make figured bass numbers.

Music types accepted:

Section 1.2.7 [bass-figure-event], page 42, and Section 1.2.52 [rest-event], page 47,

Properties (read)

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s):

Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, and Section 3.1.18 [BassFigureLine], page 389.

`Figured_bass_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.8 [FiguredBass], page 96, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.38 Figured_bass_position_engraver

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

Section 3.1.15 [BassFigureAlignmentPositioning], page 387.

`Figured_bass_position_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.39 Fingering_column_engraver

Find potentially colliding scripts and put them into a `FingeringColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.43 [FingeringColumn], page 424.

`Fingering_column_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.40 Fingering_engraver

Create fingering scripts.

Music types accepted:

Section 1.2.23 [fingering-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422.

`Fingering_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.41 Font_size_engraver

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)
: The relative size of all grobs in a context.

`Font_size_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.5 [DrumStaff], page 74, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.9 [FretBoards], page 98, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.14 [KievanStaff], page 127, Section 2.1.15 [KievanVoice], page 137, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, Section 2.1.30 [TabVoice], page 257, Section 2.1.31 [VaticanaStaff], page 270, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.42 Footnote_engraver

Create footnote texts.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.45 [FootnoteItem], page 426, and Section 3.1.46 [FootnoteSpanner], page 427.

`Footnote_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.43 Forbid_line_break_engraver

Forbid line breaks when note heads are still playing at some point.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`Forbid_line_break_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.44 Fretboard_engraver

Generate fret diagram from one or more events of type `NoteEvent`.

Music types accepted:

Section 1.2.23 [fingering-event], page 44, Section 1.2.41 [note-event], page 46, and Section 1.2.65 [string-number-event], page 49,

Properties (read)

`chordChanges` (boolean)
: Only show changes in chords scheme?

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

This engraver creates the following layout object(s):

Section 3.1.47 [FretBoard], page 428.

`Fretboard_engraver` is part of the following context(s): Section 2.1.9 [FretBoards], page 98.

2.2.45 Glissando_engraver

Engrave glissandi.

Music types accepted:

Section 1.2.25 [glissando-event], page 44,

Properties (read)

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

This engraver creates the following layout object(s):

Section 3.1.48 [Glissando], page 430.

`Glissando_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.46 Grace_auto_beam_engraver

Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or `\noBeam` will block autobeaming, just like setting the context property '`autoBeaming`' to `##f`.

Music types accepted:

Section 1.2.9 [beam-forbid-event], page 42,

Properties (read)

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Grace_auto_beam_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.47 Grace_beam_engraver

Handle `Beam` events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

Music types accepted:

Section 1.2.8 [beam-event], page 42,

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

This engraver creates the following layout object(s):

Section 3.1.19 [Beam], page 390.

`Grace_beam_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.48 Grace_engraver

Set font size and other properties for grace notes.

Properties (read)

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`Grace_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.49 Grace_spacing_engraver

Bookkeeping of shortest starting and playing notes in grace note runs.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.49 [GraceSpacing], page 431.

`Grace_spacing_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.50 Grid_line_span_engraver

This engraver makes cross-staff lines: It catches all normal lines and draws a single span line across them.

This engraver creates the following layout object(s):

Section 3.1.50 [GridLine], page 431.

`Grid_line_span_engraver` is not part of any context.

2.2.51 Grid_point_engraver

Generate grid points.

Properties (read)

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

This engraver creates the following layout object(s):

Section 3.1.51 [GridPoint], page 432.

`Grid_point_engraver` is not part of any context.

2.2.52 Grob_pq_engraver

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`Grob_pq_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.5 [DrumStaff], page 74, Section 2.1.6 [DrumVoice], page 81, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.14 [KievanStaff], page 127, Section 2.1.15 [KievanVoice], page 137, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.20 [NullVoice], page 180, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, Section 2.1.30 [TabVoice], page 257, Section 2.1.31 [VaticanaStaff], page 270, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.53 Horizontal_bracket_engraver

Create horizontal brackets over notes for musical analysis purposes.

Music types accepted:

Section 1.2.42 [note-grouping-event], page 46,

This engraver creates the following layout object(s):

Section 3.1.53 [HorizontalBracket], page 434, and Section 3.1.54 [HorizontalBracketText], page 435.

`Horizontal_bracket_engraver` is not part of any context.

2.2.54 Hyphen_engraver

Create lyric hyphens and distance constraints between words.

Music types accepted:

Section 1.2.27 [hyphen-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.66 [LyricHyphen], page 451, and Section 3.1.67 [LyricSpace], page 452.

`Hyphen_engraver` is part of the following context(s): Section 2.1.16 [Lyrics], page 151.

2.2.55 Instrument_name_engraver

Create a system start text for instrument or vocal names.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`vocalName` (markup)
: Name of a vocal line.

This engraver creates the following layout object(s):

Section 3.1.55 [InstrumentName], page 436.

`Instrument_name_engraver` is part of the following context(s): Section 2.1.1 [ChoirStaff], page 57, Section 2.1.5 [DrumStaff], page 74, Section 2.1.9 [FretBoards], page 98, Section 2.1.11 [GrandStaff], page 101, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.24 [PianoStaff], page 208, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.28 [StaffGroup], page 246, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.56 Instrument_switch_engraver

Create a cue text for taking instrument.

Properties (read)

`instrumentCueName` (markup)
The name to print if another instrument is to be taken.

This engraver creates the following layout object(s):

Section 3.1.56 [InstrumentSwitch], page 437.

`Instrument_switch_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.57 Keep_alive_together_engraver

This engraver collects all `Hara_kiri_group_spanner`s that are created in contexts at or below its own. These spanners are then tied together so that one will be removed only if all are removed. For example, if a `StaffGroup` uses this engraver, then the staves in the group will all be visible as long as there is a note in at least one of them.

`Keep_alive_together_engraver` is part of the following context(s): Section 2.1.24 [PianoStaff], page 208.

2.2.58 Key_engraver

Engrave a key signature.

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

Properties (read)

`createKeyOnClefChange` (boolean)
Print a key signature whenever the clef is changed.

`explicitKeySignatureVisibility` (vector)
'`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extraNatural` (boolean)
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`keyAlterationOrder` (list)
An alist that defines in what order alterations should be printed. The format is `(`*`step . alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
Last key signature before a key signature change.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

Properties (write)

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`tonic` (pitch)
: The tonic of the current scale.

This engraver creates the following layout object(s):

Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

`Key_engraver` is part of the following context(s): Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.59 Key_performer

Music types accepted:

Section 1.2.28 [key-change-event], page 44,

`Key_performer` is not part of any context.

2.2.60 Kievan_ligature_engraver

Handle `Kievan_ligature_events` by glueing Kievan heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.59 [KievanLigature], page 444.

`Kievan_ligature_engraver` is part of the following context(s): Section 2.1.15 [KievanVoice], page 137.

2.2.61 Laissez_vibrer_engraver

Create laissez vibrer items.

Music types accepted:

Section 1.2.30 [laissez-vibrer-event], page 44,

This engraver creates the following layout object(s):

Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.61 [LaissezVibrerTieColumn], page 446.

`Laissez_vibrer_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.62 Ledger_line_engraver

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

This engraver creates the following layout object(s):

Section 3.1.62 [LedgerLineSpanner], page 446.

`Ledger_line_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.63 Ligature_bracket_engraver

Handle `Ligature_events` by engraving `Ligature` brackets.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.64 [LigatureBracket], page 449.

`Ligature_bracket_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.30 [TabVoice], page 257, and Section 2.1.33 [Voice], page 293.

2.2.64 Lyric_engraver

Engrave text for lyrics.

Music types accepted:

Section 1.2.34 [lyric-event], page 45,

Properties (read)

`ignoreMelismata` (boolean)
: Ignore melismata for this Section "Lyrics" in *Internals Reference* line.

`lyricMelismaAlignment` (number)
: Alignment to use for a melisma syllable.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

This engraver creates the following layout object(s):

Section 3.1.68 [LyricText], page 453.

`Lyric_engraver` is part of the following context(s): Section 2.1.16 [Lyrics], page 151.

2.2.65 Lyric_performer

Music types accepted:

Section 1.2.34 [lyric-event], page 45,

`Lyric_performer` is not part of any context.

2.2.66 Mark_engraver

Create `RehearsalMark` objects. It puts them on top of all staves (which is taken from the property `stavesFound`). If moving this engraver to a different context, Section 2.2.113 [Staff_collecting_engraver], page 347, must move along, otherwise all marks end up on the same Y location.

Music types accepted:

Section 1.2.35 [mark-event], page 45,

Properties (read)

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

This engraver creates the following layout object(s):

Section 3.1.90 [RehearsalMark], page 477.

`Mark_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.67 Measure_counter_engraver

This engraver numbers ranges of measures, which is useful in parts as an aid for counting repeated measures. There is no requirement that the affected measures be repeated, however. The user delimits the area to receive a count with `\startMeasureCount` and `\stopMeasureCount`.

Music types accepted:

Section 1.2.36 [measure-counter-event], page 45,

Properties (read)

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

This engraver creates the following layout object(s):

Section 3.1.69 [MeasureCounter], page 454.

`Measure_counter_engraver` is not part of any context.

2.2.68 Measure_grouping_engraver

Create `MeasureGrouping` to indicate beat subdivision.

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beatStructure` (list)
: List of `baseMoments` that are combined to make beats.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

This engraver creates the following layout object(s):

Section 3.1.70 [MeasureGrouping], page 456.

`Measure_grouping_engraver` is not part of any context.

2.2.69 Melody_engraver

Create information for context dependent typesetting decisions.

This engraver creates the following layout object(s):

Section 3.1.71 [MelodyItem], page 457.

`Melody_engraver` is not part of any context.

2.2.70 Mensural_ligature_engraver

Handle `Mensural_ligature_events` by glueing special ligature heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45,

This engraver creates the following layout object(s):

Section 3.1.72 [MensuralLigature], page 457.

`Mensural_ligature_engraver` is part of the following context(s): Section 2.1.18 [MensuralVoice], page 165, and Section 2.1.23 [PetrucciVoice], page 194.

2.2.71 Merge_rests_engraver

Engraver to merge rests in multiple voices on the same staff. This works by gathering all rests at a time step. If they are all of the same length and there are at least two they are moved to the correct location as if there were one voice.

Properties (read)

`suspendRestMerging` (boolean)
: When using the Merge_rest_engraver do not merge rests when this is set to true.

`Merge_rests_engraver` is not part of any context.

2.2.72 Metronome_mark_engraver

Engrave metronome marking. This delegates the formatting work to the function in the `metronomeMarkFormatter` property. The mark is put over all staves. The staves are taken from the `stavesFound` property, which is maintained by Section 2.2.113 [Staff_collecting_engraver], page 347.

Music types accepted:

Section 1.2.68 [tempo-change-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

This engraver creates the following layout object(s):

Section 3.1.73 [MetronomeMark], page 458.

`Metronome_mark_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.73 Midi_control_change_performer

This performer listens to SetProperty events on context properties for generating MIDI control changes and prepares them for MIDI output.

Properties (read)

`midiBalance` (number)
: Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChorusLevel` (number)
: Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiExpression` (number)
: Expression control for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiPanPosition` (number)
: Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)
: Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`Midi_control_change_performer` is not part of any context.

2.2.74 Multi_measure_rest_engraver

Engrave multi-measure rests that are produced with 'R'. It reads `measurePosition` and `internalBarNumber` to determine what number to print over the Section 3.1.74 [MultiMeasureRest], page 459.

Music types accepted:

Section 1.2.38 [multi-measure-rest-event], page 45, and Section 1.2.39 [multi-measure-text-event], page 45,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

`Multi_measure_rest_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.75 New_fingering_engraver

Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

Properties (read)

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`strokeFingerOrientations` (list)
See `fingeringOrientations`.

This engraver creates the following layout object(s):

Section 3.1.42 [Fingering], page 422, Section 3.1.96 [Script], page 483, Section 3.1.112 [StringNumber], page 498, and Section 3.1.113 [StrokeFinger], page 500.

`New_fingering_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.76 Note_head_line_engraver

Engrave a line between two note heads in a staff switch if `followVoice` is set.

Properties (read)

`followVoice` (boolean)
If set, note heads are tracked across staff switches by a thin line.

This engraver creates the following layout object(s):

Section 3.1.138 [VoiceFollower], page 530.

`Note_head_line_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.77 Note_heads_engraver

Generate note heads.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`middleCPosition` (number)
The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`staffLineLayoutFunction` (procedure)
Layout of staff lines, `traditional`, or `semitone`.

This engraver creates the following layout object(s):

Section 3.1.80 [NoteHead], page 467.

`Note_heads_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.20 [NullVoice], page 180, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.78 Note_name_engraver

Print pitches as words.

Music types accepted:

Section 1.2.41 [note-event], page 46,

Properties (read)

`printOctaveNames` (boolean)
: Print octave marks for the `NoteNames` context.

This engraver creates the following layout object(s):

Section 3.1.81 [NoteName], page 468.

`Note_name_engraver` is part of the following context(s): Section 2.1.19 [NoteNames], page 178.

2.2.79 Note_performer

Music types accepted:

Section 1.2.6 [articulation-event], page 42, Section 1.2.14 [breathing-event], page 43, Section 1.2.41 [note-event], page 46, and Section 1.2.71 [tie-event], page 50,

`Note_performer` is not part of any context.

2.2.80 Note_spacing_engraver

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s):

Section 3.1.82 [NoteSpacing], page 468.

`Note_spacing_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.81 Ottava_spanner_engraver

Create a text spanner when the ottavation property changes.

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

This engraver creates the following layout object(s):

Section 3.1.83 [OttavaBracket], page 469.

`Ottava_spanner_engraver` is part of the following context(s): Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.82 Output_property_engraver

Apply a procedure to any grob acknowledged.

Music types accepted:

Section 1.2.4 [apply-output-event], page 42,

`Output_property_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.83 Page_turn_engraver

Decide where page turns are allowed to go.

Music types accepted:

Section 1.2.12 [break-event], page 42,

Properties (read)

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`Page_turn_engraver` is not part of any context.

2.2.84 Paper_column_engraver

Take care of generating columns.

This engraver decides whether a column is breakable. The default is that a column is always breakable. However, every `Bar_engraver` that does not have a barline at a certain point will set `forbidBreaks` in the score context to stop line breaks. In practice, this means that you can make a break point by creating a bar line (assuming that there are no beams or notes that prevent a break point).

Music types accepted:

Section 1.2.12 [break-event], page 42, and Section 1.2.29 [label-event], page 44,

Properties (read)

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

Properties (write)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

This engraver creates the following layout object(s):

Section 3.1.77 [NonMusicalPaperColumn], page 464, and Section 3.1.84 [PaperColumn], page 470.

`Paper_column_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.85 Parenthesis_engraver

Parenthesize objects whose music cause has the `parenthesize` property.

This engraver creates the following layout object(s):

Section 3.1.85 [ParenthesesItem], page 471.

`Parenthesis_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.86 Part_combine_engraver

Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.45 [part-combine-event], page 47,

Properties (read)

`aDueText` (markup)
: Text to print at a unisono passage.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

This engraver creates the following layout object(s):

Section 3.1.29 [CombineTextScript], page 402.

`Part_combine_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.87 Percent_repeat_engraver

Make whole measure repeats.

Music types accepted:

Section 1.2.47 [percent-event], page 47,

Properties (read)

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

This engraver creates the following layout object(s):

Section 3.1.86 [PercentRepeat], page 472, and Section 3.1.87 [PercentRepeatCounter], page 473.

`Percent_repeat_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.88 Phrasing_slur_engraver

Print phrasing slurs. Similar to Section 2.2.105 [Slur_engraver], page 345.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.49 [phrasing-slur-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.88 [PhrasingSlur], page 474.

`Phrasing_slur_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.89 Piano_pedal_align_engraver

Align piano pedal symbols and brackets.

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

This engraver creates the following layout object(s):

Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

`Piano_pedal_align_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.90 Piano_pedal_engraver

Engrave piano pedal symbols and brackets.

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

This engraver creates the following layout object(s):

Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

`Piano_pedal_engraver` is part of the following context(s): Section 2.1.7 [Dynamics], page 92, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.91 Piano_pedal_performer

Music types accepted:

Section 1.2.59 [sostenuto-event], page 48, Section 1.2.67 [sustain-event], page 50, and Section 1.2.77 [una-corda-event], page 51,

`Piano_pedal_performer` is not part of any context.

2.2.92 Pitch_squash_engraver

Set the vertical position of note heads to `squashedPosition`, if that property is set. This can be used to make a single-line staff demonstrating the rhythm of a melody.

Properties (read)

`squashedPosition` (integer)
: Vertical position of squashing for Section "Pitch_squash_engraver" in *Internals Reference*.

`Pitch_squash_engraver` is part of the following context(s): Section 2.1.20 [NullVoice], page 180, and Section 2.1.25 [RhythmicStaff], page 210.

2.2.93 Pitched_trill_engraver

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s):

Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, and Section 3.1.129 [TrillPitchHead], page 520.

`Pitched_trill_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.94 Pure_from_neighbor_engraver

Coordinates items that get their pure heights from their neighbors.

`Pure_from_neighbor_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.16 [Lyrics], page 151, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.95 Repeat_acknowledge_engraver

Acknowledge repeated music, and convert the contents of `repeatCommands` into an appropriate setting for `whichBar`.

Properties (read)

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where *x* is a string or `#f`. `'end-repeat` is also accepted as a command.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is '`S.|:`'.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

`Repeat_acknowledge_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.96 Repeat_tie_engraver

Create repeat ties.

Music types accepted:

Section 1.2.51 [repeat-tie-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.92 [RepeatTie], page 480, and Section 3.1.93 [RepeatTieColumn], page 481.

`Repeat_tie_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.97 Rest_collision_engraver

Handle collisions of rests.

Properties (read)

`busyGrobs` (list)
: A queue of `(`*`end-moment . grob`*`)` cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

This engraver creates the following layout object(s):

Section 3.1.95 [RestCollision], page 483.

`Rest_collision_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.98 Rest_engraver

Engrave rests.

Music types accepted:

Section 1.2.52 [rest-event], page 47,

Properties (read)

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

This engraver creates the following layout object(s):

Section 3.1.94 [Rest], page 481.

`Rest_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.99 Rhythmic_column_engraver

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s):

Section 3.1.79 [NoteColumn], page 466.

`Rhythmic_column_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.100 Script_column_engraver

Find potentially colliding scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s):

Section 3.1.97 [ScriptColumn], page 484.

`Script_column_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.101 Script_engraver

Handle note scripted articulations.

Music types accepted:

Section 1.2.6 [articulation-event], page 42,

Properties (read)

`scriptDefinitions` (list)
: The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

This engraver creates the following layout object(s):

Section 3.1.96 [Script], page 483.

`Script_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.102 Script_row_engraver

Determine order in horizontal side position elements.

This engraver creates the following layout object(s):

Section 3.1.98 [ScriptRow], page 484.

`Script_row_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.103 Separating_line_group_engraver

Generate objects for computing spacing parameters.

Properties (read)

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

Properties (write)

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

This engraver creates the following layout object(s):

Section 3.1.106 [StaffSpacing], page 492.

`Separating_line_group_engraver` is part of the following context(s): Section 2.1.2 [ChordNames], page 58, Section 2.1.5 [DrumStaff], page 74, Section 2.1.8 [FiguredBass], page 96, Section 2.1.9 [FretBoards], page 98, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.19 [NoteNames], page 178, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.104 Slash_repeat_engraver

Make beat repeats.

Music types accepted:

Section 1.2.50 [repeat-slash-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

`Slash_repeat_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.105 Slur_engraver

Build slur grobs from slur events.

Music types accepted:

Section 1.2.41 [note-event], page 46, and Section 1.2.56 [slur-event], page 48,

Properties (read)

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`slurMelismaBusy` (boolean)
: Signal if a slur is present.

This engraver creates the following layout object(s):

Section 3.1.99 [Slur], page 485.

`Slur_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.20 [NullVoice], page 180, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, and Section 2.1.33 [Voice], page 293.

2.2.106 Slur_performer

Music types accepted:

Section 1.2.56 [slur-event], page 48,

`Slur_performer` is not part of any context.

2.2.107 Spacing_engraver

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted:

Section 1.2.60 [spacing-section-event], page 48,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s):

Section 3.1.102 [SpacingSpanner], page 489.

`Spacing_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.108 Span_arpeggio_engraver

Make arpeggios that span multiple staves.

Properties (read)

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

This engraver creates the following layout object(s):

Section 3.1.9 [Arpeggio], page 379.

`Span_arpeggio_engraver` is part of the following context(s): Section 2.1.11 [GrandStaff], page 101, Section 2.1.24 [PianoStaff], page 208, and Section 2.1.28 [StaffGroup], page 246.

2.2.109 Span_bar_engraver

Make cross-staff bar lines: It catches all normal bar lines and draws a single span bar across them.

This engraver creates the following layout object(s):

Section 3.1.103 [SpanBar], page 490.

`Span_bar_engraver` is part of the following context(s): Section 2.1.11 [GrandStaff], page 101, Section 2.1.24 [PianoStaff], page 208, and Section 2.1.28 [StaffGroup], page 246.

2.2.110 Span_bar_stub_engraver

Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s):

Section 3.1.104 [SpanBarStub], page 491.

`Span_bar_stub_engraver` is part of the following context(s): Section 2.1.11 [GrandStaff], page 101, Section 2.1.24 [PianoStaff], page 208, and Section 2.1.28 [StaffGroup], page 246.

2.2.111 Span_stem_engraver

Connect cross-staff stems to the stems above in the system

This engraver creates the following layout object(s):

Section 3.1.109 [Stem], page 494.

`Span_stem_engraver` is not part of any context.

2.2.112 Spanner_break_forbid_engraver

Forbid breaks in certain spanners.

`Spanner_break_forbid_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.113 Staff_collecting_engraver

Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`Staff_collecting_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.26 [Score], page 214, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.114 Staff_performer

`Staff_performer` is not part of any context.

2.2.115 Staff_symbol_engraver

Create the constellation of five (default) staff lines.

Music types accepted:

Section 1.2.63 [staff-span-event], page 49,

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Staff_symbol_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.14 [KievanStaff], page 127, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, Section 2.1.29 [TabStaff], page 248, and Section 2.1.31 [VaticanaStaff], page 270.

2.2.116 Stanza_number_align_engraver

This engraver ensures that stanza numbers are neatly aligned.

`Stanza_number_align_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.117 Stanza_number_engraver

Engrave stanza numbers.

Properties (read)

`stanza` (markup)
: Stanza 'number' to print before the start of a verse. Use in `Lyrics` context.

This engraver creates the following layout object(s):

Section 3.1.108 [StanzaNumber], page 493.

`Stanza_number_engraver` is part of the following context(s): Section 2.1.16 [Lyrics], page 151.

2.2.118 Stem_engraver

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted:

Section 1.2.73 [tremolo-event], page 50, and Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

This engraver creates the following layout object(s):

Section 3.1.44 [Flag], page 425, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, and Section 3.1.111 [StemTremolo], page 497.

`Stem_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, and Section 2.1.33 [Voice], page 293.

2.2.119 System_start_delimiter_engraver

Create a system start delimiter (i.e., a `SystemStartBar`, `SystemStartBrace`, `SystemStartBracket` or `SystemStartSquare` spanner).

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s):

Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

`System_start_delimiter_engraver` is part of the following context(s): Section 2.1.1 [ChoirStaff], page 57, Section 2.1.11 [GrandStaff], page 101, Section 2.1.24 [PianoStaff], page 208, Section 2.1.26 [Score], page 214, and Section 2.1.28 [StaffGroup], page 246.

2.2.120 Tab_note_heads_engraver

Generate one or more tablature note heads from event of type `NoteEvent`.

Music types accepted:

Section 1.2.23 [fingering-event], page 44, Section 1.2.41 [note-event], page 46, and Section 1.2.65 [string-number-event], page 49,

Properties (read)

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

This engraver creates the following layout object(s):

Section 3.1.121 [TabNoteHead], page 508.

`Tab_note_heads_engraver` is part of the following context(s): Section 2.1.30 [TabVoice], page 257.

2.2.121 Tab_staff_symbol_engraver

Create a tablature staff symbol, but look at `stringTunings` for the number of lines.

Properties (read)

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

This engraver creates the following layout object(s):

Section 3.1.107 [StaffSymbol], page 493.

`Tab_staff_symbol_engraver` is part of the following context(s): Section 2.1.29 [TabStaff], page 248.

2.2.122 Tab_tie_follow_engraver

Adjust TabNoteHead properties when a tie is followed by a slur or glissando.

`Tab_tie_follow_engraver` is part of the following context(s): Section 2.1.30 [TabVoice], page 257.

2.2.123 Tempo_performer

Properties (read)

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`Tempo_performer` is not part of any context.

2.2.124 Text_engraver

Create text scripts.

Music types accepted:

Section 1.2.69 [text-script-event], page 50,

This engraver creates the following layout object(s):

Section 3.1.122 [TextScript], page 510.

`Text_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.125 Text_spanner_engraver

Create text spanner from an event.

Music types accepted:

Section 1.2.70 [text-span-event], page 50,

Properties (read)

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.123 [TextSpanner], page 512.

`Text_spanner_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.7 [Dynamics], page 92, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, and Section 2.1.33 [Voice], page 293.

2.2.126 Tie_engraver

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

This engraver creates the following layout object(s):

Section 3.1.124 [Tie], page 513, and Section 3.1.125 [TieColumn], page 515.

`Tie_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.19 [NoteNames], page 178, Section 2.1.20 [NullVoice], page 180, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.127 Tie_performer

Generate ties between note heads of equal pitch.

Music types accepted:

Section 1.2.71 [tie-event], page 50,

Properties (read)

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

`Tie_performer` is not part of any context.

2.2.128 Time_signature_engraver

Create a Section 3.1.126 [TimeSignature], page 515, whenever `timeSignatureFraction` changes.

Music types accepted:

Section 1.2.72 [time-signature-event], page 50,

Properties (read)

`initialTimeSignatureVisibility` (vector)
: break visibility for the initial time signature.

`partialBusy` (boolean)
: Signal that \partial acts at the current timestep.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

This engraver creates the following layout object(s):

Section 3.1.126 [TimeSignature], page 515.

`Time_signature_engraver` is part of the following context(s): Section 2.1.5 [DrumStaff], page 74, Section 2.1.12 [GregorianTranscriptionStaff], page 103, Section 2.1.17 [MensuralStaff], page 154, Section 2.1.22 [PetrucciStaff], page 183, Section 2.1.25 [RhythmicStaff], page 210, Section 2.1.27 [Staff], page 235, and Section 2.1.29 [TabStaff], page 248.

2.2.129 Time_signature_performer

`Time_signature_performer` is not part of any context.

2.2.130 Timing_translator

This engraver adds the alias `Timing` to its containing context. Responsible for synchronizing timing information from staves. Normally in `Score`. In order to create polyrhythmic music, this engraver should be removed from `Score` and placed in `Staff`.

Properties (read)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

Properties (write)

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`Timing_translator` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.131 Trill_spanner_engraver

Create trill spanner from an event.

Music types accepted:

Section 1.2.75 [trill-span-event], page 50,

Properties (read)

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s):

Section 3.1.130 [TrillSpanner], page 521.

`Trill_spanner_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.132 Tuplet_engraver

Catch tuplet events and generate appropriate bracket.

Music types accepted:

Section 1.2.76 [tuplet-span-event], page 51,

Properties (read)

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

This engraver creates the following layout object(s):

Section 3.1.131 [TupletBracket], page 522, and Section 3.1.132 [TupletNumber], page 524.

`Tuplet_engraver` is part of the following context(s): Section 2.1.3 [CueVoice], page 60, Section 2.1.6 [DrumVoice], page 81, Section 2.1.13 [GregorianTranscriptionVoice], page 114, Section 2.1.15 [KievanVoice], page 137, Section 2.1.18 [MensuralVoice], page 165, Section 2.1.23 [PetrucciVoice], page 194, Section 2.1.30 [TabVoice], page 257, Section 2.1.32 [VaticanaVoice], page 281, and Section 2.1.33 [Voice], page 293.

2.2.133 Tweak_engraver

Read the `tweaks` property from the originating event, and set properties.

`Tweak_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.2.134 Vaticana_ligature_engraver

Handle ligatures by glueing special ligature heads together.

Music types accepted:

Section 1.2.32 [ligature-event], page 45, and Section 1.2.48 [pes-or-flexa-event], page 47,

This engraver creates the following layout object(s):

Section 3.1.33 [DotColumn], page 412, and Section 3.1.135 [VaticanaLigature], page 527.

`Vaticana_ligature_engraver` is part of the following context(s): Section 2.1.32 [VaticanaVoice], page 281.

2.2.135 Vertical_align_engraver

Catch groups (staves, lyrics lines, etc.) and stack them vertically.

Properties (read)

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

This engraver creates the following layout object(s):

Section 3.1.136 [VerticalAlignment], page 528.

`Vertical_align_engraver` is part of the following context(s): Section 2.1.1 [ChoirStaff], page 57, Section 2.1.11 [GrandStaff], page 101, Section 2.1.24 [PianoStaff], page 208, Section 2.1.26 [Score], page 214, and Section 2.1.28 [StaffGroup], page 246.

2.2.136 Volta_engraver

Make volta brackets.

Properties (read)

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where x is a string or `#f`. `'end-repeat` is also accepted as a command.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

This engraver creates the following layout object(s):

Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

`Volta_engraver` is part of the following context(s): Section 2.1.26 [Score], page 214.

2.3 Tunable context properties

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`additionalBassStrings` (list)
: The additional tablature bass-strings, which will not get a seprate line in TabStaff. It is a list of the pitches of each string (starting with the lowest numbered one).

`additionalPitchPrefix` (string)
: Text with which to prefix additional pitches within a chord name.

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBassFigureAccidentals` (boolean)
: If true, then the accidentals are aligned in bass figure context.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`alternativeNumberingStyle` (symbol)
: The style of an alternative's bar numbers. Can be `numbers` for going back to the same number or `numbers-with-letters` for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

`alternativeRestores` (symbol list)
: Timing variables that are restored to their value at the start of the first alternative in subsequent alternatives.

`associatedVoice` (string)
: Name of the context (see `associatedVoiceType` for its type, usually `Voice`) that has the melody for this `Lyrics` line.

`associatedVoiceType` (symbol)
: Type of the context that has the melody for this `Lyrics` line.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" in *Internals Reference* then all staves share accidentals, and if *context* is Section "Staff" in *Internals Reference* then all voices in the same staff share accidentals, but staves do not.

  procedure The procedure represents an accidental rule to be applied to the previously specified context.

  The procedure takes the following arguments:

  `context` The current context to which the rule should be applied.

  `pitch` The pitch of the note to be evaluated.

  `barnum` The current bar number.

  `measurepos`
  : The current measure position.

  The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoBeamCheck` (procedure)
: A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-`#f` return value starts or stops the auto beam.

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`automaticBars` (boolean)
: If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
: If set to true a bar line is drawn after each note.

`barCheckSynchronize` (boolean)
: If true then reset `measurePosition` when finding a bar check.

`barNumberFormatter` (procedure)
: A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
: A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

  The following procedures are predefined:

  `all-bar-numbers-visible`
  : Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

  `first-bar-number-invisible`
  : Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

  `first-bar-number-invisible-save-broken-bars`
  : Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

  `first-bar-number-invisible-and-no-parenthesized-bar-numbers`
  : Enable bar numbers for all bars except the first bar and broken bars. This is the default.

  `(every-nth-bar-number-visible` *n*`)`
  : Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

  `(modulo-bar-number-visible` *n m*`)`
  : If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`bassFigureFormatFunction` (procedure)
: A procedure that is called to produce the formatting for a `BassFigure` grob. It takes a list of `BassFigureEvent`s, a context, and the grob to format.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*chord* `.` *markup*`)` entries.

`chordNameExceptionsFull` (list)
: An alist of full chord exceptions. Contains `(`*chord* `.` *markup*`)` entries.

`chordNameExceptionsPartial` (list)
: An alist of partial chord exceptions. Contains `(`*chord* `. (`*prefix-markup suffix-markup*`))` entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNameLowercaseMinor` (boolean)
: Downcase roots of minor chords?

`chordNameSeparator` (markup)
: The markup object used to separate parts of a chord name.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)
: The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`completionFactor` (an exact rational or procedure)
: When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.

  If `#f`, the completion engraver uses the scale-factor of each duration being split.

  If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

`completionUnit` (moment)
: Sub-bar unit of completion.

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Section "Timing_translator" in *Internals Reference* at Section "Score" in *Internals Reference* level.

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)
: A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`agostini-drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list (*`notehead-style script vertical-position`*) as values.

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`figuredBassPlusDirection` (direction)
: Where to put plus signs relative to the main figure.

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)
: If true, create a new clef when starting a staff.

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

`fontSize` (number)
: The relative size of all grobs in a context.

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`handleNegativeFrets` (symbol)
: How the automatic fret calculator should handle calculated negative frets. Values include `'ignore`, to leave them out of the diagram completely, `'include`, to include them as calculated, and `'recalculate`, to ignore the specified string and find a string where they will fit with a positive fret number.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`ignoreBarChecks` (boolean)
: Ignore bar checks.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`ignoreMelismata` (boolean)
: Ignore melismata for this Section "Lyrics" in *Internals Reference* line.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Section "Lyrics" in *Internals Reference*.

`initialTimeSignatureVisibility` (vector)
: break visibility for the initial time signature.

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(`*`min . max`*`)` pair of numbers for the loudness range of the instrument.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`instrumentTransposition` (pitch)
: Define the transposition of the instrument. Its value is the pitch that sounds when the instrument plays written middle C. This is used to transpose the MIDI output, and `\quote`s.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step . alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . `*`alter`*`)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (number)
: Alignment to use for a melisma syllable.

`magnifyStaffValue` (positive number)
: The most recent value set with `\magnifyStaff`.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`melismaBusyProperties` (list)
: A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example, if set to `'(melismaBusy beamMelismaBusy)`, only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiBalance` (number)
: Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChannelMapping` (symbol)
: How to map MIDI channels: per `staff` (default), `instrument` or `voice`.

`midiChorusLevel` (number)
: Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiExpression` (number)
: Expression control for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMergeUnisons` (boolean)
: If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`midiPanPosition` (number)
: Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)
: Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`minorChordModifier` (markup)
: Markup displayed following the root for a minor chord

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`nullAccidentals` (boolean)
: The `Accidental_engraver` generates no accidentals for notes in contexts were this is set. In addition to supressing the printed accidental, this option removes any effect the note would have had on accidentals in other voices.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`partCombineForced` (symbol)
: Override for the partcombine decision. Can be `apart`, `chords`, `unisono`, `solo1`, or `solo2`.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printOctaveNames` (boolean)
: Print octave marks for the `NoteNames` context.

`printPartCombineTexts` (boolean)
: Set ‘Solo’ and ‘A due’ texts in the part combiner?

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where x is a string or `#f`. `'end-repeat` is also accepted as a command.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`restrainOpenStrings` (boolean)
: Exclude open strings from the automatic fret calculator.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`shapeNoteStyles` (vector)
: Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`skipBars` (boolean)
: If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`slashChordSeparator` (markup)
: The markup object used to separate a chord name from its root note in case of inversions or slash chords.

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

`squashedPosition` (integer)
: Vertical position of squashing for Section "Pitch_squash_engraver" in *Internals Reference*.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

`stanza` (markup)
: Stanza 'number' to print before the start of a verse. Use in `Lyrics` context.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is '`S.|:`'.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`strictBeatBeaming` (boolean)
: Should partial beams reflect the beat structure even if it causes flags to hang out?

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

`suggestAccidentals` (boolean)
: If set, accidentals are typeset as cautionary suggestions over the note.

`supportNonIntegerFret` (boolean)
: If set in `Score` the `TabStaff` will print micro-tones as '$2\frac{1}{2}$'

`suspendRestMerging` (boolean)
: When using the Merge_rest_engraver do not merge rests when this is set to true.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`timeSignatureSettings` (list)
: A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for `baseMoment`, `beatStructure`, and `beamExceptions`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`tonic` (pitch)
: The tonic of the current scale.

`topLevelAlignment` (boolean)
: If true, the *Vertical_align_engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

`tupletSpannerDuration` (moment)
: Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

```
{
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

`vocalName` (markup)
: Name of a vocal line.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

2.4 Internal context properties

`associatedVoiceContext` (context)
: The context object of the `Voice` that has the melody for this `Lyrics`.

`barCheckLastFail` (moment)
: Where in the measure did the last barcheck fail?

`beamMelismaBusy` (boolean)
: Signal if a beam is present.

`busyGrobs` (list)
: A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g. note heads, spanners, etc.).

`currentCommandColumn` (graphical (layout) object)
: Grob that is X-parent to all current breakable (clef, key signature, etc.) items.

`currentMusicalColumn` (graphical (layout) object)
: Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

`dynamicAbsoluteVolumeFunction` (procedure)
: A procedure that takes one argument, the text value of a dynamic event, and returns the absolute volume of that dynamic event.

`finalizations` (list)
: A list of expressions to evaluate before proceeding to next time step. This is an internal variable.

`graceSettings` (list)
: Overrides for grace notes. This property should be manipulated through the `add-grace-property` function.

`hasAxisGroup` (boolean)
: True if the current context is contained in an axis group.

`hasStaffSpacing` (boolean)
: True if the current `CommandColumn` contains items that will affect spacing.

`lastChord` (markup)
: Last chord, used for detecting chord changes.

`lastKeyAlterations` (list)
: Last key signature before a key signature change.

`localAlterations` (list)
: The key signature at this point in the measure. The format is the same as for `keyAlterations`, but can also contain ((*octave* . *name*) . (*alter barnumber* . *measureposition*)) pairs.

`melismaBusy` (boolean)
: Signifies whether a melisma is active. This can be used to signal melismas on top of those automatically detected.

`partialBusy` (boolean)
: Signal that \partial acts at the current timestep.

`quotedCueEventTypes` (list)
: A list of symbols, representing the event types that should be duplicated for `\cueDuring` commands.

`quotedEventTypes` (list)
: A list of symbols, representing the event types that should be duplicated for `\quoteDuring` commands. This is also a fallback for `\cueDuring` if `quotedCueEventTypes` is not set

`rootSystem` (graphical (layout) object)
: The System object.

`scriptDefinitions` (list)
: The description of scripts. This is used by the `Script_engraver` for typesetting note-superscripts and subscripts. See `scm/script.scm` for more information.

`slurMelismaBusy` (boolean)
: Signal if a slur is present.

`stavesFound` (list of grobs)
: A list of all staff-symbols found.

`tieMelismaBusy` (boolean)
: Signal whether a tie is present.

3 Backend

3.1 All layout objects

3.1.1 Accidental

Accidental objects are created by: Section 2.2.1 [Accidental_engraver], page 306.

Standard settings:

`after-line-breaking` (boolean):
`ly:accidental-interface::remove-tied`

Dummy property, used to trigger callback for `after-line-breaking`.

`alteration` (number):
`accidental-interface::calc-alteration`

Alteration numbers for accidental.

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`extra-spacing-width` (pair of numbers):
`'(-0.2 . 0.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`glyph-name` (string):
`accidental-interface::glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list):

```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
  (1/2 . "accidentals.sharp")
  (1 . "accidentals.doublesharp")
  (-1 . "accidentals.flatflat")
  (3/4
   .
   "accidentals.sharp.slashslash.stemstemstem")
  (1/4 . "accidentals.sharp.slashslash.stem")
  (-1/4 . "accidentals.mirroredflat")
  (-3/4 . "accidentals.mirroredflat.flat"))
```

An alist of key-string pairs.

`horizontal-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::horizontal-skylines> >`
Two skylines, one to the left and one to the right of this grob.

`stencil` (stencil):
`ly:accidental-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:grob::x-parent-positioning`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.1 [accidental-interface], page 534, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.50 [inline-accidental-interface], page 566, and Section 3.2.52 [item-interface], page 568.

3.1.2 AccidentalCautionary

AccidentalCautionary objects are created by: Section 2.2.1 [Accidental_engraver], page 306.

Standard settings:

`after-line-breaking` (boolean):
`ly:accidental-interface::remove-tied`
Dummy property, used to trigger callback for `after-line-breaking`.

`alteration` (number):
`accidental-interface::calc-alteration`
Alteration numbers for accidental.

`avoid-slur` (symbol):
`'inside`
Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`glyph-name-alist` (list):
```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
```

```
(1/2 . "accidentals.sharp")
(1 . "accidentals.doublesharp")
(-1 . "accidentals.flatflat")
(3/4
 .
 "accidentals.sharp.slashslash.stemstemstem")
(1/4 . "accidentals.sharp.slashslash.stem")
(-1/4 . "accidentals.mirroredflat")
(-3/4 . "accidentals.mirroredflat.flat"))
```

An alist of key-string pairs.

`parenthesized` (boolean):

`#t`

Parenthesize this grob.

`stencil` (stencil):

`ly:accidental-interface::print`

The symbol to print.

`X-offset` (number):

`ly:grob::x-parent-positioning`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):

`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.1 [accidental-interface], page 534, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.50 [inline-accidental-interface], page 566, and Section 3.2.52 [item-interface], page 568.

3.1.3 AccidentalPlacement

AccidentalPlacement objects are created by: Section 2.2.1 [Accidental_engraver], page 306, and Section 2.2.2 [Ambitus_engraver], page 308.

Standard settings:

`direction` (direction):

`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`right-padding` (dimension, in staff space):

`0.15`

Space to insert on the right side of an object (e.g., between note and its accidentals).

`script-priority` (number):

`-100`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.2 [accidental-placement-interface], page 535, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.4 AccidentalSuggestion

AccidentalSuggestion objects are created by: Section 2.2.1 [Accidental_engraver], page 306.

Standard settings:

`after-line-breaking` (boolean):
`ly:accidental-interface::remove-tied`
Dummy property, used to trigger callback for `after-line-breaking`.

`alteration` (number):
`accidental-interface::calc-alteration`
Alteration numbers for accidental.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-size` (number):
`-2`
The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`glyph-name-alist` (list):
```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
  (1/2 . "accidentals.sharp")
  (1 . "accidentals.doublesharp")
  (-1 . "accidentals.flatflat")
  (3/4
   .
   "accidentals.sharp.slashslash.stemstemstem")
  (1/4 . "accidentals.sharp.slashslash.stem")
  (-1/4 . "accidentals.mirroredflat")
  (-3/4 . "accidentals.mirroredflat.flat"))
```
An alist of key-string pairs.

`outside-staff-priority` (number):
`0`
If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`script-priority` (number):
`0`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`0.25`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:accidental-interface::print`

The symbol to print.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.1 [accidental-interface], page 534, Section 3.2.3 [accidental-suggestion-interface], page 535, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.99 [script-interface], page 589, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.104 [side-position-interface], page 593.

3.1.5 Ambitus

Ambitus objects are created by: Section 2.2.2 [Ambitus_engraver], page 308.

Standard settings:

`axes` (list):
`'(0 1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`break-align-symbol` (symbol):
`'ambitus`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
```
'((cue-end-clef extra-space . 0.5)
  (clef extra-space . 0.5)
  (cue-clef extra-space . 0.5)
  (key-signature extra-space . 0.0)
  (staff-bar extra-space . 0.0)
  (time-signature extra-space . 0.0)
  (first-note fixed-space . 0.0))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.5 [ambitus-interface], page 536, Section 3.2.7 [axis-group-interface], page 537, Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.6 AmbitusAccidental

AmbitusAccidental objects are created by: Section 2.2.2 [Ambitus_engraver], page 308.

Standard settings:

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP=1`, `DOWN=-1`, `LEFT=-1`, `RIGHT=1`, `CENTER=0`.

`glyph-name-alist` (list):

```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
  (1/2 . "accidentals.sharp")
  (1 . "accidentals.doublesharp")
  (-1 . "accidentals.flatflat")
  (3/4
   .
   "accidentals.sharp.slashslash.stemstemstem")
  (1/4 . "accidentals.sharp.slashslash.stem")
  (-1/4 . "accidentals.mirroredflat")
  (-3/4 . "accidentals.mirroredflat.flat"))
```

An alist of key-string pairs.

`padding` (dimension, in staff space):

`0.5`

Add this much extra space between objects that are next to each other.

`side-axis` (number):

`0`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):

`ly:accidental-interface::print`

The symbol to print.

`X-offset` (number):

`ly:grob::x-parent-positioning`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):

`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.1 [accidental-interface], page 534, Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.104 [side-position-interface], page 593.

3.1.7 AmbitusLine

AmbitusLine objects are created by: Section 2.2.2 [Ambitus_engraver], page 308.

Standard settings:

`gap` (dimension, in staff space):

`ambitus-line::calc-gap`

Size of a gap in a variable symbol.

`length-fraction` (number):
`0.7`

Multiplier for lengths. Used for determining ledger lines and stem lengths.

`maximum-gap` (number):
`0.45`

Maximum value allowed for `gap` property.

`stencil` (stencil):
`ambitus::print`

The symbol to print.

`thickness` (number):
`2`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-offset` (number):
`ly:self-alignment-interface::centered-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.5 [ambitus-interface], page 536, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.8 AmbitusNoteHead

AmbitusNoteHead objects are created by: Section 2.2.2 [Ambitus_engraver], page 308.

Standard settings:

`duration-log` (integer):
`2`

The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`glyph-name` (string):
`note-head::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`ignore-ambitus` (boolean):
`#t`

If set, don't consider this notehead for ambitus calculation.

`stencil` (stencil):
`ly:note-head::print`

The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.5 [ambitus-interface], page 536, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.57 [ledgered-interface], page 571, Section 3.2.77 [note-head-interface], page 580, Section 3.2.97 [rhythmic-head-interface], page 588, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.9 Arpeggio

Arpeggio objects are created by: Section 2.2.3 [Arpeggio_engraver], page 308, and Section 2.2.108 [Span_arpeggio_engraver], page 346.

Standard settings:

`direction` (direction):
`-1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`line-thickness` (number):
`1`
For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`padding` (dimension, in staff space):
`0.5`
Add this much extra space between objects that are next to each other.

`positions` (pair of numbers):
`ly:arpeggio::calc-positions`
Pair of staff coordinates `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`protrusion` (number):
`0.4`
In an arpeggio bracket, the length of the horizontal edges.

`script-priority` (number):
`0`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`side-axis` (number):
`0`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-position` (number):
`0.0`

Vertical position, measured in half staff spaces, counted from the middle line.

`stencil` (stencil):
`ly:arpeggio::print`

The symbol to print.

`thickness` (number):
`1`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-extent` (pair of numbers):
`ly:arpeggio::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> #<primitive-procedure ly:arpeggio::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.6 [arpeggio-interface], page 536, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.10 BalloonTextItem

BalloonTextItem objects are created by: Section 2.2.6 [Balloon_engraver], page 310.

Standard settings:

`annotation-balloon` (boolean):
`#t`

Print the balloon around an annotation.

`annotation-line` (boolean):
`#t`

Print the line from an annotation to the grob that it annotates.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`stencil` (stencil):
`ly:balloon-interface::print`

The symbol to print.

`text` (markup):
`#<procedure #f (grob)>`

Text markup. See Section "Formatting text" in *Notation Reference*.

`X-offset` (number):
`#<procedure #f (grob)>`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<procedure #f (grob)>`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.8 [balloon-interface], page 540, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.125 [text-interface], page 608.

3.1.11 BarLine

BarLine objects are created by: Section 2.2.7 [Bar_engraver], page 310.

Standard settings:

`allow-span-bar` (boolean):
`#t`

If false, no inter-staff bar line will be created below this bar line.

`bar-extent` (pair of numbers):
`ly:bar-line::calc-bar-extent`

The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`break-align-anchor` (number):
`ly:bar-line::calc-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-symbol` (symbol):
`'staff-bar`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`bar-line::calc-break-visibility`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::account-for-span-bar`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`gap` (dimension, in staff space):
`0.4`

Size of a gap in a variable symbol.

`glyph` (string):
`"|"`

A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string):
`bar-line::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`hair-thickness` (number):
`1.9`

Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`kern` (dimension, in staff space):
`3.0`

The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *Staff*.StaffSymbol.thickness).

`layer` (integer):
`0`

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`rounded` (boolean)
Decide whether lines should be drawn rounded or not.

`segno-kern` (number):
`3.0`

The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *Staff*.StaffSymbol.thickness).

`space-alist` (list):

```
'((time-signature extra-space . 0.75)
  (custos minimum-space . 2.0)
  (clef extra-space . 1.0)
  (key-signature extra-space . 1.0)
  (key-cancellation extra-space . 1.0)
  (first-note fixed-space . 1.3)
  (next-note semi-fixed-space . 0.9)
  (right-edge extra-space . 0.0))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *break-align-symbol* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
used when the grob is just left of the first note on a line

`next-note`
used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
used when the grob is the last item on the line (only compatible with the `extra-space` spac-

ing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:bar-line::print`
The symbol to print.

`thick-thickness` (number):
`6.0`
Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.9 [bar-line-interface], page 540, Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.93 [pure-from-neighbor-interface], page 587.

3.1.12 BarNumber

BarNumber objects are created by: Section 2.2.8 [Bar_number_engraver], page 310.

Standard settings:

`after-line-breaking` (boolean):
`ly:side-position-interface::move-to-extremal-staff`

Dummy property, used to trigger callback for `after-line-breaking`.

`break-align-symbols` (list):
`'(left-edge staff-bar)`

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-family` (symbol):
`'roman`

The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-size` (number):
`-2`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`horizon-padding` (number):
`0.05`

The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`outside-staff-priority` (number):
`100`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`1.0`

Add this much extra space between objects that are next to each other.

`self-alignment-X` (number):
`1`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`X-offset` (number):
`self-alignment-interface::self-aligned-on-breakable`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.14 [break-alignable-interface], page 544, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.13 BassFigure

BassFigure objects are created by: Section 2.2.37 [Figured_bass_engraver], page 322.

Standard settings:

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.11 [bass-figure-interface], page 541, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.96 [rhythmic-grob-interface], page 588, and Section 3.2.125 [text-interface], page 608.

3.1.14 BassFigureAlignment

BassFigureAlignment objects are created by: Section 2.2.37 [Figured_bass_engraver], page 322.

Standard settings:

`axes` (list):
`'(1)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`padding` (dimension, in staff space):
`0.2`
Add this much extra space between objects that are next to each other.

`stacking-dir` (direction):
`-1`
Stack objects in which direction?

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.4 [align-interface], page 535, Section 3.2.7 [axis-group-interface], page 537, Section 3.2.10 [bass-figure-alignment-interface], page 541, Section 3.2.45 [grob-interface], page 559, and Section 3.2.111 [spanner-interface], page 599.

3.1.15 BassFigureAlignmentPositioning

BassFigureAlignmentPositioning objects are created by: Section 2.2.38 [Figured_bass_position_engraver], page 323.

Standard settings:

`axes` (list):
`'(1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`1.0`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.16 BassFigureBracket

BassFigureBracket objects are created by: Section 2.2.37 [Figured_bass_engraver], page 322.

Standard settings:

`edge-height` (pair):
`'(0.2 . 0.2)`
A pair of numbers specifying the heights of the vertical edges: `(left-height . right-height)`.

`stencil` (stencil):
`ly:enclosing-bracket::print`
The symbol to print.

`X-extent` (pair of numbers):
`ly:enclosing-bracket::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.30 [enclosing-bracket-interface], page 551, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.17 BassFigureContinuation

BassFigureContinuation objects are created by: Section 2.2.37 [Figured_bass_engraver], page 322.

Standard settings:

`stencil` (stencil):
`ly:figured-bass-continuation::print`
The symbol to print.

`Y-offset` (number):
`ly:figured-bass-continuation::center-on-figures`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.32 [figured-bass-continuation-interface], page 552, Section 3.2.45 [grob-interface], page 559, and Section 3.2.111 [spanner-interface], page 599.

3.1.18 BassFigureLine

BassFigureLine objects are created by: Section 2.2.37 [Figured_bass_engraver], page 322.

Standard settings:

`axes` (list):
`'(1)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`vertical-skylines` (pair of skylines):
`ly:axis-group-interface::calc-skylines`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

  Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.83 [outside-staff-axis-group-interface], page 582, and Section 3.2.111 [spanner-interface], page 599.

3.1.19 Beam

Beam objects are created by: Section 2.2.4 [Auto_beam_engraver], page 308, Section 2.2.10 [Beam_engraver], page 312, Section 2.2.16 [Chord_tremolo_engraver], page 314, Section 2.2.46 [Grace_auto_beam_engraver], page 325, and Section 2.2.47 [Grace_beam_engraver], page 326.

Standard settings:

`auto-knee-gap` (dimension, in staff space):
: `5.5`

  If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

`beam-thickness` (dimension, in staff space):
: `0.48`

  Beam thickness, measured in `staff-space` units.

`beamed-stem-shorten` (list):
: `'(1.0 0.5 0.25)`

  How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair):
: `ly:beam::calc-beaming`

  Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`clip-edges` (boolean):
: `#t`

  Allow outward pointing beamlets at the edges of beams?

`collision-interfaces` (list):
: ```
  '(beam-interface
     clef-interface
     clef-modifier-interface
     flag-interface
     inline-accidental-interface
     key-signature-interface
     note-head-interface
     stem-interface
     time-signature-interface)
  ```

  A list of interfaces for which automatic beam-collision resolution is run.
  ```

`damping` (number):
`1`

Amount of beam slope damping.

`details` (list):

```
'((secondary-beam-demerit . 10)
  (stem-length-demerit-factor . 5)
  (region-size . 2)
  (beam-eps . 0.001)
  (stem-length-limit-penalty . 5000)
  (damping-direction-penalty . 800)
  (hint-direction-penalty . 20)
  (musical-direction-factor . 400)
  (ideal-slope-factor . 10)
  (collision-penalty . 500)
  (collision-padding . 0.35)
  (round-to-zero-slope . 0.02))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:beam::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-family` (symbol):
`'roman`

The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`gap` (dimension, in staff space):
`0.8`

Size of a gap in a variable symbol.

`neutral-direction` (direction):
`-1`

Which direction to take in the center of the staff.

`normalized-endpoints` (pair):
`ly:spanner::calc-normalized-endpoints`

Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`positions` (pair of numbers):
`beam::place-broken-parts-individually`

Pair of staff coordinates `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`stencil` (stencil):
`ly:beam::print`

The symbol to print.

`transparent` (boolean):
`#<procedure #f (grob)>`

This makes the grob invisible.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

`X-positions` (pair of numbers):
`ly:beam::calc-x-positions`

Pair of X staff coordinates of a spanner in the form `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff.

This object supports the following interface(s): Section 3.2.12 [beam-interface], page 542, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, Section 3.2.115 [staff-symbol-referencer-interface], page 602, and Section 3.2.134 [unbreakable-spanner-interface], page 616.

3.1.20 BendAfter

BendAfter objects are created by: Section 2.2.12 [Bend_engraver], page 312.

Standard settings:

`minimum-length` (dimension, in staff space):
`0.5`

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`stencil` (stencil):
`bend::print`

The symbol to print.

`thickness` (number):
`2.0`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

This object supports the following interface(s): Section 3.2.13 [bend-after-interface], page 544, Section 3.2.45 [grob-interface], page 559, and Section 3.2.111 [spanner-interface], page 599.

3.1.21 BreakAlignGroup

BreakAlignGroup objects are created by: Section 2.2.13 [Break_align_engraver], page 313.

Standard settings:

`axes` (list):
`'(0)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-average-anchor`
Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-visibility` (vector):
`ly:break-aligned-interface::calc-break-visibility`
A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.22 BreakAlignment

BreakAlignment objects are created by: Section 2.2.13 [Break_align_engraver], page 313.

Standard settings:

`axes` (list):
`'(0)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`break-align-orders` (vector):

```
#((left-edge
    cue-end-clef
    ambitus
    breathing-sign
    clef
    cue-clef
    staff-bar
    key-cancellation
    key-signature
    time-signature
    custos)
  (left-edge
    cue-end-clef
    ambitus
    breathing-sign
```

```
    clef
    cue-clef
    staff-bar
    key-cancellation
    key-signature
    time-signature
    custos)
  (left-edge
    ambitus
    breathing-sign
    clef
    key-cancellation
    key-signature
    time-signature
    staff-bar
    cue-clef
    custos))
```

This is a vector of 3 lists: `#(`*`end-of-line unbroken start-of-line`*`)`. Each list contains *break-align symbols* that specify an order of breakable items (see Section "break-alignment-interface" in *Internals Reference*).

For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
                    ambitus
                    breathing-sign
                    time-signature
                    clef
                    cue-clef
                    staff-bar
                    key-cancellation
                    key-signature
                    custos))
```

`non-musical` (boolean):

`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`stacking-dir` (direction):

`1`

Stack objects in which direction?

`X-extent` (pair of numbers):

`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.16 [break-alignment-interface], page 546, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.23 BreathingSign

BreathingSign objects are created by: Section 2.2.14 [Breathing_sign_engraver], page 313.

Standard settings:

`break-align-symbol` (symbol):

`'breathing-sign`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):

`#(#t #t #f)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`non-musical` (boolean):

`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):

```
'((ambitus extra-space . 2.0)
  (custos minimum-space . 1.0)
  (key-signature minimum-space . 1.5)
  (time-signature minimum-space . 1.5)
  (staff-bar minimum-space . 1.5)
  (clef minimum-space . 2.0)
  (cue-clef minimum-space . 2.0)
  (cue-end-clef minimum-space . 2.0)
  (first-note fixed-space . 1.0)
  (right-edge extra-space . 0.1))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
: `ly:text-interface::print`

  The symbol to print.

`text` (markup):
: `'(#<procedure musicglyph-markup (layout props glyph-name)>`
  `"scripts.rcomma")`

  Text markup. See Section "Formatting text" in *Notation Reference*.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure`
  `ly:grob::stencil-height> >`

  Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `ly:breathing-sign::offset-callback`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.17 [breathing-sign-interface], page 547, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, and Section 3.2.125 [text-interface], page 608.

3.1.24 ChordName

ChordName objects are created by: Section 2.2.15 [Chord_name_engraver], page 313.

Standard settings:

`after-line-breaking` (boolean):
: `ly:chord-name::after-line-breaking`

Dummy property, used to trigger callback for `after-line-breaking`.

`extra-spacing-height` (pair of numbers):
`'(0.2 . -0.2)`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(-0.5 . 0.5)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-family` (symbol):
`'sans`

The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-size` (number):
`1.5`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`word-space` (dimension, in staff space):
`0.0`

Space to insert between words in texts.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.18 [chord-name-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.96 [rhythmic-grob-interface], page 588, and Section 3.2.125 [text-interface], page 608.

3.1.25 Clef

Clef objects are created by: Section 2.2.17 [Clef_engraver], page 314.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number):
`1`

Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-symbol` (symbol):
`'clef`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-at-beginning-of-line`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`glyph-name` (string):
`ly:clef::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):

```
'((cue-clef extra-space . 2.0)
  (staff-bar extra-space . 0.7)
  (key-cancellation minimum-space . 3.5)
  (key-signature minimum-space . 3.5)
  (time-signature minimum-space . 4.2)
```

```
(first-note minimum-fixed-space . 5.0)
(next-note extra-space . 1.0)
(right-edge extra-space . 0.5))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *break-align-symbol* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:clef::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.19 [clef-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.26 ClefModifier

ClefModifier objects are created by: Section 2.2.17 [Clef_engraver], page 314, and Section 2.2.24 [Cue_clef_engraver], page 317.

Standard settings:

`break-visibility` (vector):
`#<procedure #f (grob)>`
A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`clef-alignments` (list):
`'((G -0.2 . 0.1) (F -0.3 . -0.2) (C 0 . 0))`
An alist of parent-alignments that should be used for clef modifiers with various clefs

`color` (color):
`#<procedure #f (grob)>`
The color of this grob.

`font-shape` (symbol):
`'italic`
Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number):
`-4`
The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12%

larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`parent-alignment-X` (number):
`ly:clef-modifier::calc-parent-alignment`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`staff-padding` (dimension, in staff space):
`0.7`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`transparent` (boolean):
`#<procedure #f (grob)>`

This makes the grob invisible.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.20 [clef-modifier-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559,

Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.27 ClusterSpanner

ClusterSpanner objects are created by: Section 2.2.18 [Cluster_spanner_engraver], page 315.

Standard settings:

`minimum-length` (dimension, in staff space):
`0.0`
Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`padding` (dimension, in staff space):
`0.25`
Add this much extra space between objects that are next to each other.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:cluster::print`
The symbol to print.

`style` (symbol):
`'ramp`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

This object supports the following interface(s): Section 3.2.22 [cluster-interface], page 549, Section 3.2.45 [grob-interface], page 559, and Section 3.2.111 [spanner-interface], page 599.

3.1.28 ClusterSpannerBeacon

ClusterSpannerBeacon objects are created by: Section 2.2.18 [Cluster_spanner_engraver], page 315.

Standard settings:

`Y-extent` (pair of numbers):
`ly:cluster-beacon::height`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.21 [cluster-beacon-interface], page 548, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.96 [rhythmic-grob-interface], page 588.

3.1.29 CombineTextScript

CombineTextScript objects are created by: Section 2.2.86 [Part_combine_engraver], page 339.

Standard settings:

`avoid-slur` (symbol):
`'outside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`baseline-skip` (dimension, in staff space):
`2`

Distance between base lines of multiple lines of text.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-series` (symbol):
`'bold`

Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`outside-staff-priority` (number):
`450`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)
Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`script-priority` (number):
`200`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
: `1`

  If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
: `0.5`

  Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
: `ly:text-interface::print`

  The symbol to print.

`X-offset` (number):
: `ly:self-alignment-interface::aligned-on-x-parent`

  The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

  Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.125 [text-interface], page 608, and Section 3.2.126 [text-script-interface], page 609.

3.1.30 CueClef

CueClef objects are created by: Section 2.2.24 [Cue_clef_engraver], page 317.

Standard settings:

`avoid-slur` (symbol):
: `'inside`

  Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-symbol` (symbol):
`'cue-clef`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-at-beginning-of-line`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`font-size` (number):
`-4`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`full-size-change` (boolean):
`#t`

Don't make a change clef smaller.

`glyph-name` (string):
`ly:clef::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
```
'((staff-bar minimum-space . 2.7)
  (key-cancellation minimum-space . 3.5)
  (key-signature minimum-space . 3.5)
  (time-signature minimum-space . 4.2)
  (custos minimum-space . 0.0)
```

```
(first-note minimum-fixed-space . 3.0)
(next-note extra-space . 1.0)
(right-edge extra-space . 0.5))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:clef::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.19 [clef-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.31 CueEndClef

CueEndClef objects are created by: Section 2.2.24 [Cue_clef_engraver], page 317.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-symbol` (symbol):
`'cue-end-clef`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#t #t #f)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-at-beginning-of-line`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`font-size` (number):
`-4`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`full-size-change` (boolean):
`#t`

Don't make a change clef smaller.

`glyph-name` (string):
`ly:clef::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
```
'((clef extra-space . 0.7)
  (cue-clef extra-space . 0.7)
  (staff-bar extra-space . 0.7)
  (key-cancellation minimum-space . 3.5)
  (key-signature minimum-space . 3.5)
  (time-signature minimum-space . 4.2)
  (first-note minimum-fixed-space . 5.0)
  (next-note extra-space . 1.0)
  (right-edge extra-space . 0.5))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
: `ly:clef::print`

  The symbol to print.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

  Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.19 [clef-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.32 Custos

Custos objects are created by: Section 2.2.25 [Custos_engraver], page 317.

Standard settings:

`break-align-symbol` (symbol):
: `'custos`

  This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
: `#(#t #f #f)`

  A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`neutral-direction` (direction):
: `-1`

  Which direction to take in the center of the staff.

`non-musical` (boolean):
: `#t`

  True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
: `'((first-note minimum-fixed-space . 0.0)`
  `(right-edge extra-space . 0.1))`

  An alist that specifies distances from this grob to other breakable items, using the format:

  ```
  '((break-align-symbol . (spacing-style . space))
    (break-align-symbol . (spacing-style . space))
    ...)
  ```

  Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

  `first-note`
  : used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:custos::print`

The symbol to print.

`style` (symbol):
`'vaticana`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.23 [custos-interface], page 549, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.33 DotColumn

DotColumn objects are created by: Section 2.2.27 [Dot_column_engraver], page 318, and Section 2.2.134 [Vaticana_ligature_engraver], page 354.

Standard settings:

`axes` (list):
`'(0)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`chord-dots-limit` (integer):
`3`
Limits the column of dots on each chord to the height of the chord plus `chord-dots-limit` staff-positions.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.24 [dot-column-interface], page 549, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.34 Dots

Dots objects are created by: Section 2.2.28 [Dots_engraver], page 319.

Standard settings:

`avoid-slur` (symbol):
`'inside`
Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`dot-count` (integer):
`dots::calc-dot-count`
The number of dots.

`extra-spacing-height` (pair of numbers):
`'(-0.5 . 0.5)`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(0.0 . 0.2)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`staff-position` (number):
`dots::calc-staff-position`

Vertical position, measured in half staff spaces, counted from the middle line.

`stencil` (stencil):
`ly:dots::print`

The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.25 [dots-interface], page 550, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.35 DoublePercentRepeat

DoublePercentRepeat objects are created by: Section 2.2.29 [Double_percent_repeat_engraver], page 319.

Standard settings:

`break-align-symbol` (symbol):
`'staff-bar`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#t #t #f)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`dot-negative-kern` (number):
`0.75`

The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`font-encoding` (symbol):
`'fetaMusic`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`slash-negative-kern` (number):
`1.6`

The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number):
`1.0`

The slope of this object.

`stencil` (stencil):
`ly:percent-repeat-item-interface::double-percent`

The symbol to print.

`thickness` (number):
`0.48`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.87 [percent-repeat-interface], page 585, and Section 3.2.88 [percent-repeat-item-interface], page 585.

3.1.36 DoublePercentRepeatCounter

DoublePercentRepeatCounter objects are created by: Section 2.2.29 [Double_percent_repeat_engraver], page 319.

Standard settings:

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or

DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

font-encoding (symbol):
'fetaText

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are fetaMusic (Emmentaler), fetaBraces, fetaText (Emmentaler).

font-size (number):
-2

The font size, compared to the 'normal' size. 0 is style-sheet's normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property fontSize is set, its value is added to this before the glyph is printed. Fractional values are allowed.

padding (dimension, in staff space):
0.2

Add this much extra space between objects that are next to each other.

parent-alignment-X (number):
0

Specify on which point of the parent the object is aligned. The value -1 means aligned on parent's left edge, 0 on center, and 1 right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from self-alignment-X property will be used.

self-alignment-X (number):
0

Specify alignment of an object. The value -1 means left aligned, 0 centered, and 1 right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

side-axis (number):
1

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

staff-padding (dimension, in staff space):
0.25

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

stencil (stencil):
ly:text-interface::print

The symbol to print.

X-offset (number):
ly:self-alignment-interface::aligned-on-x-parent

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.87 [percent-repeat-interface], page 585, Section 3.2.88 [percent-repeat-item-interface], page 585, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.37 DoubleRepeatSlash

DoubleRepeatSlash objects are created by: Section 2.2.104 [Slash_repeat_engraver], page 345.

Standard settings:

`dot-negative-kern` (number):
`0.75`
The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`font-encoding` (symbol):
`'fetaMusic`
The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`slash-negative-kern` (number):
`1.6`
The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number):
`1.0`
The slope of this object.

`stencil` (stencil):
`ly:percent-repeat-item-interface::beat-slash`
The symbol to print.

`thickness` (number):
`0.48`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current

staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.87 [percent-repeat-interface], page 585, Section 3.2.88 [percent-repeat-item-interface], page 585, and Section 3.2.96 [rhythmic-grob-interface], page 588.

3.1.38 DynamicLineSpanner

DynamicLineSpanner objects are created by: Section 2.2.32 [Dynamic_align_engraver], page 320.

Standard settings:

`axes` (list):
`'(1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`minimum-space` (dimension, in staff space):
`1.2`

Minimum distance that the victim should move (after padding).

`outside-staff-priority` (number):
`250`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.6`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`slur-padding` (number):
`0.3`

Extra distance between slur and script.

`staff-padding` (dimension, in staff space):
`0.1`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-element-stencils> #<primitive-procedure ly:grob::pure-vertical-skylines-from-element-stencils> >`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.26 [dynamic-interface], page 550, Section 3.2.27 [dynamic-line-spanner-interface], page 551, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.39 DynamicText

DynamicText objects are created by: Section 2.2.33 [Dynamic_engraver], page 320.

Standard settings:

`direction` (direction):
`ly:script-interface::calc-direction`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`
In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-encoding` (symbol):
`'fetaText`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-series` (symbol):
`'bold`

Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol):
`'italic`

Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`right-padding` (dimension, in staff space):
`0.5`

Space to insert on the right side of an object (e.g., between note and its accidentals).

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`X-align-on-main-noteheads` (boolean):
`#t`

If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<procedure #f (grob)> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.26 [dynamic-interface], page 550, Section 3.2.28 [dynamic-text-interface], page 551, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.99 [script-interface], page 589, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.125 [text-interface], page 608.

3.1.40 DynamicTextSpanner

DynamicTextSpanner objects are created by: Section 2.2.33 [Dynamic_engraver], page 320.

Standard settings:

`before-line-breaking` (boolean):
`dynamic-text-spanner::before-line-breaking`

Dummy property, used to trigger a callback function.

`bound-details` (list):
```
'((right (attach-dir . -1)
         (Y . 0)
         (padding . 0.75))
  (right-broken (attach-dir . 1) (padding . 0.0))
  (left (attach-dir . -1)
        (Y . 0)
        (stencil-offset -0.75 . -0.5)
        (padding . 0.75))
  (left-broken (attach-dir . 1)))
```

An alist of properties for determining attachments of spanners to edges.

`dash-fraction` (number):
`0.2`

Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number):
`3.0`

The length of one dash together with whitespace. If negative, no line is drawn at all.

`font-shape` (symbol):
`'italic`

Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number):
`1`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`left-bound-info` (list):
`ly:line-spanner::calc-left-bound-info-and-text`

An alist of properties for determining attachments of spanners to edges.

`minimum-length` (dimension, in staff space):
`2.0`

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-Y-extent` (pair of numbers):
`'(-1 . 1)`

Minimum size of an object in Y dimension, measured in `staff-space` units.

`right-bound-info` (list):
`ly:line-spanner::calc-right-bound-info`

An alist of properties for determining attachments of spanners to edges.

`skyline-horizontal-padding` (number):
`0.2`

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`

Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:line-spanner::print`

The symbol to print.

`style` (symbol):
`'dashed-line`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

This object supports the following interface(s): Section 3.2.26 [dynamic-interface], page 550, Section 3.2.29 [dynamic-text-spanner-interface], page 551, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.41 Episema

Episema objects are created by: Section 2.2.35 [Episema_engraver], page 321.

Standard settings:

`bound-details` (list):
```
'((left (Y . 0) (padding . 0) (attach-dir . -1))
  (right (Y . 0) (padding . 0) (attach-dir . 1)))
```
An alist of properties for determining attachments of spanners to edges.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`left-bound-info` (list):
`ly:line-spanner::calc-left-bound-info`
An alist of properties for determining attachments of spanners to edges.

`right-bound-info` (list):
`ly:line-spanner::calc-right-bound-info`
An alist of properties for determining attachments of spanners to edges.

`side-axis` (number):
`1`
If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:line-spanner::print`
The symbol to print.

`style` (symbol):
`'line`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.31 [episema-interface], page 552, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.42 Fingering

Fingering objects are created by: Section 2.2.40 [Fingering_engraver], page 323, and Section 2.2.75 [New_fingering_engraver], page 335.

Standard settings:

`add-stem-support` (boolean):
`only-if-beamed`

If set, the `Stem` object is included in this script's support.

`avoid-slur` (symbol):
`'around`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`direction` (direction):
`ly:script-interface::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-encoding` (symbol):
`'fetaText`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-size` (number):
`-5`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`parent-alignment-Y` (number):
`0`

Like `parent-alignment-X` but for the Y axis.

`script-priority` (number):
`100`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number):
`0`

Like `self-alignment-X` but for the Y axis.

`slur-padding` (number):
`0.2`

Extra distance between slur and script.

`staff-padding` (dimension, in staff space):
`0.5`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`text` (markup):
`fingering::calc-text`

Text markup. See Section "Formatting text" in *Notation Reference*.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.33 [finger-interface], page 552, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.125 [text-interface], page 608, and Section 3.2.126 [text-script-interface], page 609.

3.1.43 FingeringColumn

FingeringColumn objects are created by: Section 2.2.39 [Fingering_column_engraver], page 323.

Standard settings:

`padding` (dimension, in staff space):
`0.2`

Add this much extra space between objects that are next to each other.

`snap-radius` (number):
`0.3`
The maximum distance between two objects that will cause them to snap to alignment along an axis.

This object supports the following interface(s): Section 3.2.34 [fingering-column-interface], page 553, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.44 Flag

Flag objects are created by: Section 2.2.118 [Stem_engraver], page 348.

Standard settings:

`color` (color):
`#<procedure #f (grob)>`
The color of this grob.

`glyph-name` (string):
`ly:flag::glyph-name`
The glyph name within the font.
In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`stencil` (stencil):
`ly:flag::print`
The symbol to print.

`transparent` (boolean):
`#<procedure #f (grob)>`
This makes the grob invisible.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:flag::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
`ly:flag::calc-x-offset`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:flag::calc-y-offset> #<primitive-procedure ly:flag::pure-calc-y-offset> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.35 [flag-interface], page 553, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.45 FootnoteItem

FootnoteItem objects are created by: Section 2.2.42 [Footnote_engraver], page 324.

Standard settings:

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean):
: `#t`

  Print the line from an annotation to the grob that it annotates.

`automatically-numbered` (boolean):
: `#<procedure #f (grob)>`

  Should a footnote be automatically numbered?

`break-visibility` (vector):
: `#<procedure #f (grob)>`

  A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`footnote` (boolean):
: `#t`

  Should this be a footnote or in-note?

`footnote-text` (markup):
: `#<procedure #f (grob)>`

  A footnote for the grob.

`stencil` (stencil):
: `ly:balloon-interface::print`

  The symbol to print.

`text` (markup):
: `#<procedure #f (grob)>`

  Text markup. See Section “Formatting text” in *Notation Reference*.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

`X-offset` (number):
: `#<procedure #f (grob)>`

  The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

`Y-offset` (number):
: `#<procedure #f (grob)>`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.8 [balloon-interface], page 540, Section 3.2.36 [font-interface], page 553, Section 3.2.37 [footnote-interface], page 555, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.125 [text-interface], page 608.

3.1.46 FootnoteSpanner

FootnoteSpanner objects are created by: Section 2.2.42 [Footnote_engraver], page 324.

Standard settings:

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean):
: `#t`

  Print the line from an annotation to the grob that it annotates.

`automatically-numbered` (boolean):
: `#<procedure #f (grob)>`

  Should a footnote be automatically numbered?

`footnote` (boolean):
: `#t`

  Should this be a footnote or in-note?

`footnote-text` (markup):
: `#<procedure #f (grob)>`

  A footnote for the grob.

`stencil` (stencil):
: `ly:balloon-interface::print-spanner`

  The symbol to print.

`text` (markup):
: `#<procedure #f (grob)>`

  Text markup. See Section “Formatting text” in *Notation Reference*.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
: `#<procedure #f (grob)>`

  The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<procedure #f (grob)>`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.8 [balloon-interface], page 540, Section 3.2.36 [font-interface], page 553, Section 3.2.37 [footnote-interface], page 555, Section 3.2.38 [footnote-spanner-interface], page 555, Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.47 FretBoard

FretBoard objects are created by: Section 2.2.44 [Fretboard_engraver], page 324.

Standard settings:

`after-line-breaking` (boolean):
`ly:chord-name::after-line-breaking`

Dummy property, used to trigger callback for `after-line-breaking`.

`extra-spacing-height` (pair of numbers):
`'(0.2 . -0.2)`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(-0.5 . 0.5)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`fret-diagram-details` (list):
`'((finger-code . below-string))`

An alist of detailed grob properties for fret diagrams. Each alist entry consists of a (*property* . *value*) pair. The properties which can be included in `fret-diagram-details` include the following:

- `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
- `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
- `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
- `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
- `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
- `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
- `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
- `fret-count` – The number of frets. Default 4.
- `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
- `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.

- `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
- `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
- `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
- `paren-padding` – The padding for the parenthesis. Default 0.05.
- `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
- `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
- `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`stencil` (stencil):

`fret-board::calc-stencil`

The symbol to print.

`Y-extent` (pair of numbers):

`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.18 [chord-name-interface], page 548, Section 3.2.36 [font-interface], page 553, Section 3.2.39 [fret-diagram-interface], page 555, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568,

Section 3.2.84 [outside-staff-interface], page 582, and Section 3.2.96 [rhythmic-grob-interface], page 588.

3.1.48 Glissando

Glissando objects are created by: Section 2.2.45 [Glissando_engraver], page 325.

Standard settings:

`after-line-breaking` (boolean):
`ly:spanner::kill-zero-spanned-time`
Dummy property, used to trigger callback for `after-line-breaking`.

`bound-details` (list):
```
'((right (attach-dir . -1)
         (end-on-accidental . #t)
         (padding . 0.5))
  (left (attach-dir . 1)
        (padding . 0.5)
        (start-at-dot . #t)))
```
An alist of properties for determining attachments of spanners to edges.

`gap` (dimension, in staff space):
`0.5`
Size of a gap in a variable symbol.

`left-bound-info` (list):
`ly:line-spanner::calc-left-bound-info`
An alist of properties for determining attachments of spanners to edges.

`normalized-endpoints` (pair):
`ly:spanner::calc-normalized-endpoints`
Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`right-bound-info` (list):
`ly:line-spanner::calc-right-bound-info`
An alist of properties for determining attachments of spanners to edges.

`simple-Y` (boolean):
`#t`
Should the Y placement of a spanner disregard changes in system heights?

`stencil` (stencil):
`ly:line-spanner::print`
The symbol to print.

`style` (symbol):
`'line`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`zigzag-width` (dimension, in staff space):
: `0.75`

  The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

This object supports the following interface(s): Section 3.2.40 [glissando-interface], page 557, Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.134 [unbreakable-spanner-interface], page 616.

3.1.49 GraceSpacing

GraceSpacing objects are created by: Section 2.2.49 [Grace_spacing_engraver], page 327.

Standard settings:

`common-shortest-duration` (moment):
: `grace-spacing::calc-shortest-duration`

  The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`shortest-duration-space` (number):
: `1.6`

  Start with this multiple of `spacing-increment` space for the shortest duration. See also Section "spacing-spanner-interface" in *Internals Reference*.

`spacing-increment` (dimension, in staff space):
: `0.8`

  The unit of length for note-spacing. Typically, the width of a note head. See also Section "spacing-spanner-interface" in *Internals Reference*.

This object supports the following interface(s): Section 3.2.41 [grace-spacing-interface], page 557, Section 3.2.45 [grob-interface], page 559, Section 3.2.108 [spacing-options-interface], page 597, and Section 3.2.111 [spanner-interface], page 599.

3.1.50 GridLine

GridLine objects are created by: Section 2.2.50 [Grid_line_span_engraver], page 327.

Standard settings:

`layer` (integer):
: `0`

  An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:grid-line-interface::print`

The symbol to print.

`X-extent` (pair of numbers):
`ly:grid-line-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.43 [grid-line-interface], page 559, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.100 [self-alignment-interface], page 590.

3.1.51 GridPoint

GridPoint objects are created by: Section 2.2.51 [Grid_point_engraver], page 327.

Standard settings:

`X-extent` (pair of numbers):
`'(0 . 0)`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`'(0 . 0)`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.44 [grid-point-interface], page 559, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.52 Hairpin

Hairpin objects are created by: Section 2.2.33 [Dynamic_engraver], page 320.

Standard settings:

`after-line-breaking` (boolean):
`ly:spanner::kill-zero-spanned-time`

Dummy property, used to trigger callback for `after-line-breaking`.

`bound-padding` (number):
`1.0`
The amount of padding to insert around spanner bounds.

`broken-bound-padding` (number):
`ly:hairpin::broken-bound-padding`
The amount of padding to insert when a spanner is broken at a line break.

`circled-tip` (boolean)
Put a circle at start/end of hairpins (al/del niente).

`grow-direction` (direction):
`hairpin::calc-grow-direction`
Crescendo or decrescendo?

`height` (dimension, in staff space):
`0.6666`
Height of an object in `staff-space` units.

`minimum-length` (dimension, in staff space):
`2.0`
Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`self-alignment-Y` (number):
`0`
Like `self-alignment-X` but for the Y axis.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:hairpin::print`
The symbol to print.

`thickness` (number):
`1.0`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`to-barline` (boolean):
`#t`
If true, the spanner will stop at the bar line just before it would otherwise stop.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> #<primitive-procedure ly:hairpin::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:self-alignment-interface::y-aligned-on-self> #<primitive-procedure ly:self-alignment-interface::pure-y-aligned-on-self> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.26 [dynamic-interface], page 550, Section 3.2.45 [grob-interface], page 559, Section 3.2.46 [hairpin-interface], page 563, Section 3.2.61 [line-interface], page 572, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.111 [spanner-interface], page 599.

3.1.53 HorizontalBracket

HorizontalBracket objects are created by: Section 2.2.53 [Horizontal_bracket_engraver], page 328.

Standard settings:

`bracket-flare` (pair of numbers):
`'(0.5 . 0.5)`

A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`connect-to-neighbor` (pair):
`ly:tuplet-bracket::calc-connect-to-neighbors`

Pair of booleans, indicating whether this grob looks as a continued break.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space):
`0.2`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`0.2`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:horizontal-bracket::print`

The symbol to print.

`thickness` (number):
`1.0`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.48 [horizontal-bracket-interface], page 565, Section 3.2.61 [line-interface], page 572, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.54 HorizontalBracketText

HorizontalBracketText objects are created by: Section 2.2.53 [Horizontal_bracket_engraver], page 328.

Standard settings:

`direction` (direction):
`ly:horizontal-bracket-text::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-size` (number):
`-1`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:horizontal-bracket-text::print`

The symbol to print.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.49 [horizontal-bracket-text-interface], page 565, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.55 InstrumentName

InstrumentName objects are created by: Section 2.2.55 [Instrument_name_engraver], page 328.

Standard settings:

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space):
`0.3`

Add this much extra space between objects that are next to each other.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number):
`0`

Like `self-alignment-X` but for the Y axis.

`stencil` (stencil):
`system-start-text::print`

The symbol to print.

`X-offset` (number):
`system-start-text::calc-x-offset`

The horizontal amount that this object is moved relative to its X-parent.

`Y-offset` (number):
`system-start-text::calc-y-offset`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, Section 3.2.123 [system-start-text-interface], page 607, and Section 3.2.125 [text-interface], page 608.

3.1.56 InstrumentSwitch

InstrumentSwitch objects are created by: Section 2.2.56 [Instrument_switch_engraver], page 329.

Standard settings:

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`outside-staff-priority` (number):
`500`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)
: Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
: `-1`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
: `1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
: `0.5`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
: `ly:text-interface::print`

The symbol to print.

`X-offset` (number):
: `ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.57 KeyCancellation

KeyCancellation objects are created by: Section 2.2.58 [Key_engraver], page 329.

Standard settings:

`break-align-symbol` (symbol):
: `'key-cancellation`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#t #t #f)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-including-staff`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(0.0 . 1.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flat-positions` (list):
`'(2 3 4 2 1 2 1)`

Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`glyph-name-alist` (list):
`'((0 . "accidentals.natural"))`

An alist of key-string pairs.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`sharp-positions` (list):
`'(4 5 4 2 3 2 3)`

Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`space-alist` (list):

```
'((time-signature extra-space . 1.25)
  (staff-bar extra-space . 0.6)
  (key-signature extra-space . 0.5)
  (cue-clef extra-space . 0.5)
```

```
(right-edge extra-space . 0.5)
(first-note fixed-space . 2.5)
(custos extra-space . 1.0))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *break-align-symbol* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:key-signature-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.53 [key-cancellation-interface], page 570, Section 3.2.54 [key-signature-interface], page 570, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.58 KeySignature

KeySignature objects are created by: Section 2.2.58 [Key_engraver], page 329.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number):
`1`

Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-symbol` (symbol):
`'key-signature`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-including-staff`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(0.0 . 1.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flat-positions` (list):
`'(2 3 4 2 1 2 1)`

Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`glyph-name-alist` (list):

```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
  (1/2 . "accidentals.sharp")
  (1 . "accidentals.doublesharp")
  (-1 . "accidentals.flatflat")
  (3/4
   .
   "accidentals.sharp.slashslash.stemstemstem")
  (1/4 . "accidentals.sharp.slashslash.stem")
  (-1/4 . "accidentals.mirroredflat")
  (-3/4 . "accidentals.mirroredflat.flat"))
```

An alist of key-string pairs.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`sharp-positions` (list):
`'(4 5 4 2 3 2 3)`

Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`space-alist` (list):

```
'((time-signature extra-space . 1.15)
  (staff-bar extra-space . 1.1)
  (cue-clef extra-space . 0.5)
  (right-edge extra-space . 0.5)
  (first-note fixed-space . 2.5))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
`ly:key-signature-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.54 [key-signature-interface], page 570, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.59 KievanLigature

KievanLigature objects are created by: Section 2.2.60 [Kievan_ligature_engraver], page 330.

Standard settings:

`padding` (dimension, in staff space):
`0.5`
Add this much extra space between objects that are next to each other.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:kievan-ligature::print`
The symbol to print.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.55 [kievan-ligature-interface], page 570, and Section 3.2.111 [spanner-interface], page 599.

3.1.60 LaissezVibrerTie

LaissezVibrerTie objects are created by: Section 2.2.61 [Laissez_vibrer_engraver], page 330.

Standard settings:

`control-points` (list of number pairs):
`ly:semi-tie::calc-control-points`

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list):
`'((ratio . 0.333) (height-limit . 1.0))`

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:tie::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-height` (pair of numbers):
`'(-0.5 . 0.5)`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`head-direction` (direction):
`-1`

Are the note heads left or right in a semitie?

`stencil` (stencil):
`laissez-vibrer::print`

The symbol to print.

`thickness` (number):
`1.0`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.102 [semi-tie-interface], page 591.

3.1.61 LaissezVibrerTieColumn

LaissezVibrerTieColumn objects are created by: Section 2.2.61 [Laissez_vibrer_engraver], page 330.

Standard settings:

`head-direction` (direction):
`ly:semi-tie-column::calc-head-direction`
Are the note heads left or right in a semitie?

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.101 [semi-tie-column-interface], page 591.

3.1.62 LedgerLineSpanner

LedgerLineSpanner objects are created by: Section 2.2.62 [Ledger_line_engraver], page 331.

Standard settings:

`layer` (integer):
`0`
An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`length-fraction` (number):
`0.25`
Multiplier for lengths. Used for determining ledger lines and stem lengths.

`minimum-length-fraction` (number):
`0.25`
Minimum length of ledger line as fraction of note head size.

`springs-and-rods` (boolean):
`ly:ledger-line-spanner::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:ledger-line-spanner::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.56 [ledger-line-spanner-interface], page 571, and Section 3.2.111 [spanner-interface], page 599.

3.1.63 LeftEdge

LeftEdge objects are created by: Section 2.2.13 [Break_align_engraver], page 313.

Standard settings:

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`
Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-symbol` (symbol):
`'left-edge`
This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #f #t)`
A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`non-musical` (boolean):
`#t`
True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
```
'((ambitus extra-space . 2.0)
  (breathing-sign minimum-space . 0.0)
  (cue-end-clef extra-space . 0.8)
  (clef extra-space . 0.8)
```

```
(cue-clef extra-space . 0.8)
(staff-bar extra-space . 0.0)
(key-cancellation extra-space . 0.0)
(key-signature extra-space . 0.8)
(time-signature extra-space . 1.0)
(custos extra-space . 0.0)
(first-note fixed-space . 2.0)
(right-edge extra-space . 0.0))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *break-align-symbol* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`X-extent` (pair of numbers):
`'(0 . 0)`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`'(0 . 0)`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.64 LigatureBracket

LigatureBracket objects are created by: Section 2.2.63 [Ligature_bracket_engraver], page 331.

Standard settings:

`bracket-visibility` (boolean or symbol):
`#t`

This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`connect-to-neighbor` (pair):
`ly:tuplet-bracket::calc-connect-to-neighbors`

Pair of booleans, indicating whether this grob looks as a continued break.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair):
`'(0.7 . 0.7)`

A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`padding` (dimension, in staff space):
`2.0`

Add this much extra space between objects that are next to each other.

`positions` (pair of numbers):
`ly:tuplet-bracket::calc-positions`

Pair of staff coordinates (*left . right*), where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`shorten-pair` (pair of numbers):
`'(-0.2 . -0.2)`

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`staff-padding` (dimension, in staff space):
`0.25`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:tuplet-bracket::print`

The symbol to print.

`thickness` (number):
`1.6`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-positions` (pair of numbers):
`ly:tuplet-bracket::calc-x-positions`

Pair of X staff coordinates of a spanner in the form (*left . right*), where both *left* and *right* are in `staff-space` units of the current staff.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.132 [tuplet-bracket-interface], page 613.

3.1.65 LyricExtender

LyricExtender objects are created by: Section 2.2.36 [Extender_engraver], page 322.

Standard settings:

`minimum-length` (dimension, in staff space):
`1.5`

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`stencil` (stencil):
`ly:lyric-extender::print`

The symbol to print.

`thickness` (number):
`0.8`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-extent` (pair of numbers):
`'(0 . 0)`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.63 [lyric-extender-interface], page 574, Section 3.2.65 [lyric-interface], page 575, and Section 3.2.111 [spanner-interface], page 599.

3.1.66 LyricHyphen

LyricHyphen objects are created by: Section 2.2.54 [Hyphen_engraver], page 328.

Standard settings:

`after-line-breaking` (boolean):
`ly:spanner::kill-zero-spanned-time`

Dummy property, used to trigger callback for `after-line-breaking`.

`dash-period` (number):
`10.0`

The length of one dash together with whitespace. If negative, no line is drawn at all.

`height` (dimension, in staff space):
`0.42`

Height of an object in `staff-space` units.

`length` (dimension, in staff space):
`0.66`

User override for the stem length of unbeamed stems.

`minimum-distance` (dimension, in staff space):
`0.1`

Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space):
`0.3`

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`padding` (dimension, in staff space):
`0.07`

Add this much extra space between objects that are next to each other.

`springs-and-rods` (boolean):
`ly:lyric-hyphen::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:lyric-hyphen::print`
The symbol to print.

`thickness` (number):
`1.3`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`'(0 . 0)`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.64 [lyric-hyphen-interface], page 574, Section 3.2.65 [lyric-interface], page 575, and Section 3.2.111 [spanner-interface], page 599.

3.1.67 LyricSpace

LyricSpace objects are created by: Section 2.2.54 [Hyphen_engraver], page 328.

Standard settings:

`minimum-distance` (dimension, in staff space):
`0.45`
Minimum distance between rest and notes or beam.

`padding` (dimension, in staff space):
`0.0`
Add this much extra space between objects that are next to each other.

`springs-and-rods` (boolean):
`ly:lyric-hyphen::set-spacing-rods`
Dummy variable for triggering spacing routines.

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.64 [lyric-hyphen-interface], page 574, and Section 3.2.111 [spanner-interface], page 599.

3.1.68 LyricText

LyricText objects are created by: Section 2.2.64 [Lyric_engraver], page 331.

Standard settings:

`extra-spacing-height` (pair of numbers):
`'(0.2 . -0.2)`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(0.0 . 0.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-series` (symbol):
`'medium`

Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-size` (number):
`1.0`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`parent-alignment-X` (number):
`'()`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`skyline-horizontal-padding` (number):
`0.1`

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs

from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`stencil` (stencil):
`lyric-text::print`
The symbol to print.

`text` (markup):
`#<procedure #f (grob)>`
Text markup. See Section "Formatting text" in *Notation Reference*.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`word-space` (dimension, in staff space):
`0.6`
Space to insert between words in texts.

`X-align-on-main-noteheads` (boolean):
`#t`
If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.66 [lyric-syllable-interface], page 575, Section 3.2.96 [rhythmic-grob-interface], page 588, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.125 [text-interface], page 608.

3.1.69 MeasureCounter

MeasureCounter objects are created by: Section 2.2.67 [Measure_counter_engraver], page 332.

Standard settings:

`count-from` (integer):
`1`
The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or

DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

font-encoding (symbol):
'fetaText

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are fetaMusic (Emmentaler), fetaBraces, fetaText (Emmentaler).

font-size (number):
-2

The font size, compared to the 'normal' size. 0 is style-sheet's normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property fontSize is set, its value is added to this before the glyph is printed. Fractional values are allowed.

outside-staff-horizontal-padding (number):
0.5

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

outside-staff-priority (number):
750

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller outside-staff-priority is closer to the staff.

self-alignment-X (number):
0

Specify alignment of an object. The value -1 means left aligned, 0 centered, and 1 right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

side-axis (number):
1

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

spacing-pair (pair):
'(break-alignment . break-alignment)

A pair of alignment symbols which set an object's spacing relative to its left and right BreakAlignments.

For example, a MultiMeasureRest will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

staff-padding (dimension, in staff space):
0.5

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`measure-counter-stencil`

The symbol to print.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.68 [measure-counter-interface], page 575, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.70 MeasureGrouping

MeasureGrouping objects are created by: Section 2.2.68 [Measure_grouping_engraver], page 332.

Standard settings:

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`height` (dimension, in staff space):
`2.0`

Height of an object in `staff-space` units.

`padding` (dimension, in staff space):
`2`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`3`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:measure-grouping::print`

The symbol to print.

`thickness` (number):
`1`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.69 [measure-grouping-interface], page 576, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.71 MelodyItem

MelodyItem objects are created by: Section 2.2.69 [Melody_engraver], page 333.

Standard settings:

`neutral-direction` (direction):
`-1`
Which direction to take in the center of the staff.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.70 [melody-spanner-interface], page 576.

3.1.72 MensuralLigature

MensuralLigature objects are created by: Section 2.2.70 [Mensural_ligature_engraver], page 333.

Standard settings:

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`
Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:mensural-ligature::print`
The symbol to print.

`thickness` (number):
`1.3`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.71 [mensural-ligature-interface], page 576, and Section 3.2.111 [spanner-interface], page 599.

3.1.73 MetronomeMark

MetronomeMark objects are created by: Section 2.2.72 [Metronome_mark_engraver], page 333.

Standard settings:

`after-line-breaking` (boolean):
`ly:side-position-interface::move-to-extremal-staff`

Dummy property, used to trigger callback for `after-line-breaking`.

`break-align-symbols` (list):
`'(time-signature)`

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #t #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flag-style` (symbol):
`'default`

The style of the flag to be used with `MetronomeMark`. Available are `'modern-straight-flag`, `'old-straight-flag`, `flat-flag`, `mensural` and `'default`

`non-break-align-symbols` (list):
`'(paper-column-interface)`

A list of symbols that determine which NON-break-aligned interfaces to align this to.

`outside-staff-horizontal-padding` (number):
`0.2`

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-priority` (number):
`1000`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.8`
Add this much extra space between objects that are next to each other.

`self-alignment-X` (number):
`-1`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`
If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`self-alignment-interface::self-aligned-on-breakable`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.14 [break-alignable-interface], page 544, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.72 [metronome-mark-interface], page 577, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.74 MultiMeasureRest

MultiMeasureRest objects are created by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

Standard settings:

`bound-padding` (number):
`0.5`

The amount of padding to insert around spanner bounds.

`expand-limit` (integer):
`10`

Maximum number of measures expanded in church rests.

`hair-thickness` (number):
`2.0`

Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`max-symbol-separation` (number):
`8.0`

The maximum distance between symbols making up a church rest.

`round-up-exceptions` (list):
`'()`

A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`spacing-pair` (pair):
`'(break-alignment . break-alignment)`

A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

`springs-and-rods` (boolean):
`ly:multi-measure-rest::set-spacing-rods`

Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:multi-measure-rest::print`

The symbol to print.

`thick-thickness` (number):
`6.6`

Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`usable-duration-logs` (list):
`'(-3 -2 -1 0)`

List of `duration-log`s that can be used in typesetting the grob.

`voiced-position` (number):
`4`

The staff-position of a voiced `Rest`, negative if the rest has `direction DOWN`.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:multi-measure-rest::height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.73 [multi-measure-interface], page 577, Section 3.2.74 [multi-measure-rest-interface], page 577, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.95 [rest-interface], page 587, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.75 MultiMeasureRestNumber

MultiMeasureRestNumber objects are created by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

Standard settings:

`bound-padding` (number):
`1.0`

The amount of padding to insert around spanner bounds.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-encoding` (symbol):
`'fetaText`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`padding` (dimension, in staff space):
`0.4`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge,

in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`
If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`springs-and-rods` (boolean):
`ly:multi-measure-rest::set-text-rods`
Dummy variable for triggering spacing routines.

`staff-padding` (dimension, in staff space):
`0.4`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.73 [multi-measure-interface], page 577, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.76 MultiMeasureRestText

MultiMeasureRestText objects are created by: Section 2.2.74 [Multi_measure_rest_engraver], page 335.

Standard settings:

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`outside-staff-priority` (number):
`450`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.2`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`skyline-horizontal-padding` (number):
`0.2`

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`staff-padding` (dimension, in staff space):
`0.25`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.73 [multi-measure-interface], page 577, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.77 NonMusicalPaperColumn

NonMusicalPaperColumn objects are created by: Section 2.2.84 [Paper_column_engraver], page 338.

Standard settings:

`allow-loose-spacing` (boolean):
`#t`

If set, column can be detached from main spacing.

`axes` (list):
`'(0)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`before-line-breaking` (boolean):
`ly:paper-column::before-line-breaking`

Dummy property, used to trigger a callback function.

`font-size` (number):
`-7.5`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`full-measure-extra-space` (number):
`1.0`

Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`horizontal-skylines` (pair of skylines):
`ly:separation-item::calc-skylines`

Two skylines, one to the left and one to the right of this grob.

`keep-inside-line` (boolean):
`#t`

If set, this column cannot have objects sticking into the margin.

`layer` (integer):
`1000`

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`line-break-permission` (symbol):
`'allow`

Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`page-break-permission` (symbol):
`'allow`

Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.85 [paper-column-interface], page 583, Section 3.2.103 [separation-item-interface], page 592, and Section 3.2.106 [spaceable-grob-interface], page 597.

3.1.78 NoteCollision

NoteCollision objects are created by: Section 2.2.19 [Collision_engraver], page 315.

Standard settings:

`axes` (list):
`'(0 1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`note-collision-threshold` (dimension, in staff space):
`1`

Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`prefer-dotted-right` (boolean):
`#t`

For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.75 [note-collision-interface], page 578.

3.1.79 NoteColumn

NoteColumn objects are created by: Section 2.2.99 [Rhythmic_column_engraver], page 343.

Standard settings:

`axes` (list):
`'(0 1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`horizontal-skylines` (pair of skylines):
`ly:separation-item::calc-skylines`

Two skylines, one to the left and one to the right of this grob.

`skyline-vertical-padding` (number):
`0.15`

The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.76 [note-column-interface], page 579, and Section 3.2.103 [separation-item-interface], page 592.

3.1.80 NoteHead

NoteHead objects are created by: Section 2.2.20 [Completion_heads_engraver], page 315, Section 2.2.31 [Drum_notes_engraver], page 320, and Section 2.2.77 [Note_heads_engraver], page 336.

Standard settings:

`duration-log` (integer):
`note-head::calc-duration-log`
The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`extra-spacing-height` (pair of numbers):
`ly:note-head::include-ledger-line-height`
In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`glyph-name` (string):
`note-head::calc-glyph-name`
The glyph name within the font.
In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`parenthesis-friends` (list):
`'(accidental-grob dot)`
A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`stem-attachment` (pair of numbers):
`ly:note-head::calc-stem-attachment`
An `(x . y)` pair where the stem attaches to the notehead.

`stencil` (stencil):
`ly:note-head::print`
The symbol to print.

`X-offset` (number):
`ly:note-head::stem-x-shift`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.42 [gregorian-ligature-interface], page 558, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.57 [ledgered-interface], page 571, Section 3.2.59 [ligature-head-interface], page 572, Section 3.2.71 [mensural-ligature-interface], page 576, Section 3.2.77 [note-head-interface], page 580, Section 3.2.96 [rhythmic-grob-interface], page 588, Section 3.2.97 [rhythmic-head-interface], page 588, Section 3.2.115 [staff-symbol-referencer-interface], page 602, and Section 3.2.135 [vaticana-ligature-interface], page 616.

3.1.81 NoteName

NoteName objects are created by: Section 2.2.78 [Note_name_engraver], page 336.

Standard settings:

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.78 [note-name-interface], page 580, and Section 3.2.125 [text-interface], page 608.

3.1.82 NoteSpacing

NoteSpacing objects are created by: Section 2.2.80 [Note_spacing_engraver], page 337.

Standard settings:

`knee-spacing-correction` (number):
`1.0`

Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`same-direction-correction` (number):
`0.25`

Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`space-to-barline` (boolean):
`#t`
If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`stem-spacing-correction` (number):
`0.5`
Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.79 [note-spacing-interface], page 580, and Section 3.2.107 [spacing-interface], page 597.

3.1.83 OttavaBracket

OttavaBracket objects are created by: Section 2.2.81 [Ottava_spanner_engraver], page 337.

Standard settings:

`dash-fraction` (number):
`0.3`
Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair):
`'(0 . 0.8)`
A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`font-shape` (symbol):
`'italic`
Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`minimum-length` (dimension, in staff space):
`0.3`
Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`outside-staff-priority` (number):
`400`
If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.5`
Add this much extra space between objects that are next to each other.

`shorten-pair` (pair of numbers):
`'(-0.8 . -0.6)`
The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`staff-padding` (dimension, in staff space):
`2.0`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:ottava-bracket::print`
The symbol to print.

`style` (symbol):
`'dashed-line`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.48 [horizontal-bracket-interface], page 565, Section 3.2.61 [line-interface], page 572, Section 3.2.82 [ottava-bracket-interface], page 581, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.84 PaperColumn

PaperColumn objects are created by: Section 2.2.84 [Paper_column_engraver], page 338.

Standard settings:

`allow-loose-spacing` (boolean):
`#t`
If set, column can be detached from main spacing.

`axes` (list):
`'(0)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`before-line-breaking` (boolean):
`ly:paper-column::before-line-breaking`

Dummy property, used to trigger a callback function.

`font-size` (number):
`-7.5`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`horizontal-skylines` (pair of skylines):
`ly:separation-item::calc-skylines`

Two skylines, one to the left and one to the right of this grob.

`keep-inside-line` (boolean):
`#t`

If set, this column cannot have objects sticking into the margin.

`layer` (integer):
`1000`

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`skyline-vertical-padding` (number):
`0.08`

The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.85 [paper-column-interface], page 583, Section 3.2.103 [separation-item-interface], page 592, and Section 3.2.106 [spaceable-grob-interface], page 597.

3.1.85 ParenthesesItem

ParenthesesItem objects are created by: Section 2.2.85 [Parenthesis_engraver], page 339.

Standard settings:

`font-size` (number):
`-6`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):
`0.2`
Add this much extra space between objects that are next to each other.

`stencil` (stencil):
`parentheses-item::print`
The symbol to print.

`stencils` (list):
`parentheses-item::calc-parenthesis-stencils`
Multiple stencils, used as intermediate value.

`X-extent` (pair of numbers):
`'(0 . 0)`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`parentheses-item::y-extent`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.86 [parentheses-interface], page 585.

3.1.86 PercentRepeat

PercentRepeat objects are created by: Section 2.2.87 [Percent_repeat_engraver], page 339.

Standard settings:

`dot-negative-kern` (number):
`0.75`
The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`font-encoding` (symbol):
`'fetaMusic`
The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`slope` (number):
`1.0`
The slope of this object.

`spacing-pair` (pair):
`'(break-alignment . staff-bar)`
A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

`springs-and-rods` (boolean):
`ly:multi-measure-rest::set-spacing-rods`

Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:multi-measure-rest::percent`

The symbol to print.

`thickness` (number):
`0.48`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.74 [multi-measure-rest-interface], page 577, Section 3.2.87 [percent-repeat-interface], page 585, and Section 3.2.111 [spanner-interface], page 599.

3.1.87 PercentRepeatCounter

PercentRepeatCounter objects are created by: Section 2.2.87 [Percent_repeat_engraver], page 339.

Standard settings:

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-encoding` (symbol):
`'fetaText`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-size` (number):
`-2`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):
`0.2`
Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`
Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`staff-padding` (dimension, in staff space):
`0.25`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.87 [percent-repeat-interface], page 585, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.125 [text-interface], page 608.

3.1.88 PhrasingSlur

PhrasingSlur objects are created by: Section 2.2.88 [Phrasing_slur_engraver], page 340.

Standard settings:

`control-points` (list of number pairs):
`ly:slur::calc-control-points`

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list):

```
'((region-size . 4)
  (head-encompass-penalty . 1000.0)
  (stem-encompass-penalty . 30.0)
  (edge-attraction-factor . 4)
  (same-slope-penalty . 20)
  (steeper-slope-factor . 50)
  (non-horizontal-penalty . 15)
  (max-slope . 1.1)
  (max-slope-factor . 10)
  (free-head-distance . 0.3)
  (free-slur-distance . 0.8)
  (gap-to-staffline-inside . 0.2)
  (gap-to-staffline-outside . 0.1)
  (extra-object-collision-penalty . 50)
  (accidental-collision . 3)
  (extra-encompass-free-distance . 0.3)
  (extra-encompass-collision-distance . 0.8)
  (head-slur-distance-max-ratio . 3)
  (head-slur-distance-factor . 10)
  (absolute-closeness-measure . 0.3)
  (edge-slope-exponent . 1.7)
  (close-to-edge-length . 2.5)
  (encompass-object-range-overshoot . 0.5)
  (slur-tie-extrema-min-distance . 0.2)
  (slur-tie-extrema-min-distance-penalty . 2))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):

`ly:slur::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`height-limit` (dimension, in staff space):

`2.0`

Maximum slur height: The longer the slur, the closer it is to this height.

`minimum-length` (dimension, in staff space):

`1.5`

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`ratio` (number):
`0.333`

Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`

Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:slur::print`

The symbol to print.

`thickness` (number):
`1.1`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:slur::vertical-skylines> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:slur::height> #<primitive-procedure ly:slur::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.105 [slur-interface], page 594, and Section 3.2.111 [spanner-interface], page 599.

3.1.89 PianoPedalBracket

PianoPedalBracket objects are created by: Section 2.2.90 [Piano_pedal_engraver], page 340.

Standard settings:

`bound-padding` (number):
`1.0`

The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers):
`'(0.5 . 0.5)`

A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair):
`'(1.0 . 1.0)`

A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`shorten-pair` (pair of numbers):
`'(0.0 . 0.0)`

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`stencil` (stencil):
`ly:piano-pedal-bracket::print`

The symbol to print.

`style` (symbol):
`'line`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number):
`1.0`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.89 [piano-pedal-bracket-interface], page 586, Section 3.2.90 [piano-pedal-interface], page 586, and Section 3.2.111 [spanner-interface], page 599.

3.1.90 RehearsalMark

RehearsalMark objects are created by: Section 2.2.66 [Mark_engraver], page 332.

Standard settings:

`after-line-breaking` (boolean):
`ly:side-position-interface::move-to-extremal-staff`

Dummy property, used to trigger callback for `after-line-breaking`.

`baseline-skip` (dimension, in staff space):
`2`

Distance between base lines of multiple lines of text.

`break-align-symbols` (list):
`'(staff-bar key-signature clef)`

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#f #t #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-size` (number):
`2`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`outside-staff-horizontal-padding` (number):
`0.2`

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-priority` (number):
`1500`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.8`
Add this much extra space between objects that are next to each other.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`self-alignment-interface::self-aligned-on-breakable`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.14 [break-alignable-interface], page 544, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.67 [mark-interface], page 575, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.125 [text-interface], page 608.

3.1.91 RepeatSlash

RepeatSlash objects are created by: Section 2.2.104 [Slash_repeat_engraver], page 345.

Standard settings:

`slash-negative-kern` (number):
`0.85`
The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number):
`1.7`
The slope of this object.

`stencil` (stencil):
`ly:percent-repeat-item-interface::beat-slash`
The symbol to print.

`thickness` (number):
`0.48`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.87 [percent-repeat-interface], page 585, Section 3.2.88 [percent-repeat-item-interface], page 585, and Section 3.2.96 [rhythmic-grob-interface], page 588.

3.1.92 RepeatTie

RepeatTie objects are created by: Section 2.2.96 [Repeat_tie_engraver], page 342.

Standard settings:

`control-points` (list of number pairs):
`ly:semi-tie::calc-control-points`
List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list):
`'((ratio . 0.333) (height-limit . 1.0))`
Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:tie::calc-direction`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-height` (pair of numbers):
`'(-0.5 . 0.5)`
In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`head-direction` (direction):
`1`
Are the note heads left or right in a semitie?

`stencil` (stencil):
`ly:tie::print`
The symbol to print.

`thickness` (number):
`1.0`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.102 [semi-tie-interface], page 591.

3.1.93 RepeatTieColumn

RepeatTieColumn objects are created by: Section 2.2.96 [Repeat_tie_engraver], page 342.

Standard settings:

`head-direction` (direction):
`ly:semi-tie-column::calc-head-direction`
Are the note heads left or right in a semitie?

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.101 [semi-tie-column-interface], page 591.

3.1.94 Rest

Rest objects are created by: Section 2.2.21 [Completion_rest_engraver], page 316, and Section 2.2.98 [Rest_engraver], page 343.

Standard settings:

`duration-log` (integer):
`stem::calc-duration-log`
The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`minimum-distance` (dimension, in staff space):
`0.25`
Minimum distance between rest and notes or beam.

`parenthesis-friends` (list):
`'(dot)`
A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`stencil` (stencil):
`ly:rest::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`voiced-position` (number):
`4`
The staff-position of a voiced `Rest`, negative if the rest has `direction DOWN`.

`X-extent` (pair of numbers):
`ly:rest::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:rest::height> #<primitive-procedure ly:rest::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:rest::y-offset-callback> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.95 [rest-interface], page 587, Section 3.2.96 [rhythmic-grob-interface], page 588, Section 3.2.97 [rhythmic-head-interface], page 588, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.95 RestCollision

RestCollision objects are created by: Section 2.2.97 [Rest_collision_engraver], page 343.

Standard settings:

`minimum-distance` (dimension, in staff space):
`0.75`

Minimum distance between rest and notes or beam.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.94 [rest-collision-interface], page 587.

3.1.96 Script

Script objects are created by: Section 2.2.31 [Drum_notes_engraver], page 320, Section 2.2.75 [New_fingering_engraver], page 335, and Section 2.2.101 [Script_engraver], page 344.

Standard settings:

`add-stem-support` (boolean):
`#t`

If set, the `Stem` object is included in this script's support.

`direction` (direction):
`ly:script-interface::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-encoding` (symbol):
`'fetaMusic`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`horizon-padding` (number):
`0.1`

The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`slur-padding` (number):
`0.2`

Extra distance between slur and script.

`staff-padding` (dimension, in staff space):
`0.25`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:script-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`script-interface::calc-x-offset`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.99 [script-interface], page 589, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.104 [side-position-interface], page 593.

3.1.97 ScriptColumn

ScriptColumn objects are created by: Section 2.2.100 [Script_column_engraver], page 344.

Standard settings:

`before-line-breaking` (boolean):
`ly:script-column::before-line-breaking`
Dummy property, used to trigger a callback function.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.98 [script-column-interface], page 589.

3.1.98 ScriptRow

ScriptRow objects are created by: Section 2.2.102 [Script_row_engraver], page 344.

Standard settings:

`before-line-breaking` (boolean):
`ly:script-column::row-before-line-breaking`
Dummy property, used to trigger a callback function.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.98 [script-column-interface], page 589.

3.1.99 Slur

Slur objects are created by: Section 2.2.105 [Slur_engraver], page 345.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`control-points` (list of number pairs):
`ly:slur::calc-control-points`

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list):

```
'((region-size . 4)
  (head-encompass-penalty . 1000.0)
  (stem-encompass-penalty . 30.0)
  (edge-attraction-factor . 4)
  (same-slope-penalty . 20)
  (steeper-slope-factor . 50)
  (non-horizontal-penalty . 15)
  (max-slope . 1.1)
  (max-slope-factor . 10)
  (free-head-distance . 0.3)
  (free-slur-distance . 0.8)
  (gap-to-staffline-inside . 0.2)
  (gap-to-staffline-outside . 0.1)
  (extra-object-collision-penalty . 50)
  (accidental-collision . 3)
  (extra-encompass-free-distance . 0.3)
  (extra-encompass-collision-distance . 0.8)
  (head-slur-distance-max-ratio . 3)
  (head-slur-distance-factor . 10)
  (absolute-closeness-measure . 0.3)
  (edge-slope-exponent . 1.7)
  (close-to-edge-length . 2.5)
  (encompass-object-range-overshoot . 0.5)
  (slur-tie-extrema-min-distance . 0.2)
  (slur-tie-extrema-min-distance-penalty . 2))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:slur::calc-direction`

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

height-limit (dimension, in staff space):

2.0

Maximum slur height: The longer the slur, the closer it is to this height.

line-thickness (number):

0.8

For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

minimum-length (dimension, in staff space):

1.5

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the springs-and-rods property. If added to a Tie, this sets the minimum distance between noteheads.

ratio (number):

0.25

Parameter for slur shape. The higher this number, the quicker the slur attains its height-limit.

springs-and-rods (boolean):

ly:spanner::set-spacing-rods

Dummy variable for triggering spacing routines.

stencil (stencil):

ly:slur::print

The symbol to print.

thickness (number):

1.2

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

vertical-skylines (pair of skylines):

#<unpure-pure-container #<primitive-procedure ly:slur::vertical-skylines> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >

Two skylines, one above and one below this grob.

Y-extent (pair of numbers):

#<unpure-pure-container #<primitive-procedure ly:slur::height> #<primitive-procedure ly:slur::pure-height> >

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.105 [slur-interface], page 594, and Section 3.2.111 [spanner-interface], page 599.

3.1.100 SostenutoPedal

SostenutoPedal objects are created by: Section 2.2.90 [Piano_pedal_engraver], page 340.

Standard settings:

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`
In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-shape` (symbol):
`'italic`
Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`padding` (dimension, in staff space):
`0.0`
Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)
Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.91 [piano-pedal-script-interface], page 586, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.125 [text-interface], page 608.

3.1.101 SostenutoPedalLineSpanner

SostenutoPedalLineSpanner objects are created by: Section 2.2.89 [Piano_pedal_align_engraver], page 340.

Standard settings:

`axes` (list):
`'(1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`minimum-space` (dimension, in staff space):
`1.0`

Minimum distance that the victim should move (after padding).

`outside-staff-priority` (number):
`1000`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`1.2`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`1.0`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-element-stencils> #<primitive-procedure ly:grob::pure-vertical-skylines-from-element-stencils> >`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.90 [piano-pedal-interface], page 586, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.102 SpacingSpanner

SpacingSpanner objects are created by: Section 2.2.107 [Spacing_engraver], page 346.

Standard settings:

`average-spacing-wishes` (boolean):
`#t`

If set, the spacing wishes are averaged over staves.

`base-shortest-duration` (moment):
`#<Mom 3/16>`

Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`common-shortest-duration` (moment):
`ly:spacing-spanner::calc-common-shortest-duration`

The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`shortest-duration-space` (number):
`2.0`

Start with this multiple of `spacing-increment` space for the shortest duration. See also Section "spacing-spanner-interface" in *Internals Reference*.

`spacing-increment` (dimension, in staff space):
`1.2`
The unit of length for note-spacing. Typically, the width of a note head. See also Section "spacing-spanner-interface" in *Internals Reference*.

`springs-and-rods` (boolean):
`ly:spacing-spanner::set-springs`
Dummy variable for triggering spacing routines.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.108 [spacing-options-interface], page 597, Section 3.2.109 [spacing-spanner-interface], page 598, and Section 3.2.111 [spanner-interface], page 599.

3.1.103 SpanBar

SpanBar objects are created by: Section 2.2.109 [Span_bar_engraver], page 346.

Standard settings:

`allow-span-bar` (boolean):
`#t`
If false, no inter-staff bar line will be created below this bar line.

`bar-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`before-line-breaking` (boolean):
`ly:span-bar::before-line-breaking`
Dummy property, used to trigger a callback function.

`break-align-symbol` (symbol):
`'staff-bar`
This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`glyph-name` (string):
`ly:span-bar::calc-glyph-name`
The glyph name within the font.
In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`layer` (integer):
`0`
An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`non-musical` (boolean):
`#t`
True if the grob belongs to a `NonMusicalPaperColumn`.

`stencil` (stencil):
`ly:span-bar::print`
The symbol to print.

`X-extent` (pair of numbers):
`ly:span-bar::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`'(+inf.0 . -inf.0)`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.9 [bar-line-interface], page 540, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.110 [span-bar-interface], page 598.

3.1.104 SpanBarStub

SpanBarStub objects are created by: Section 2.2.110 [Span_bar_stub_engraver], page 346.

Standard settings:

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height`
In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`X-extent` (pair of numbers):
`#<procedure #f (grob)>`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #f #<procedure pure-from-neighbor-interface::pure-height (grob beg end)> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.93 [pure-from-neighbor-interface], page 587.

3.1.105 StaffGrouper

StaffGrouper objects are not created by any engraver.

Standard settings:

`staff-staff-spacing` (list):
`'((basic-distance . 9)`

```
(minimum-distance . 7)
(padding . 1)
(stretchability . 5))
```

When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

- `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list):

```
'((basic-distance . 10.5)
  (minimum-distance . 8)
  (padding . 1)
  (stretchability . 9))
```

The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.112 [staff-grouper-interface], page 600.

3.1.106 StaffSpacing

StaffSpacing objects are created by: Section 2.2.103 [Separating_line_group_engraver], page 344.

Standard settings:

`non-musical` (boolean):

```
#t
```

True if the grob belongs to a `NonMusicalPaperColumn`.

`stem-spacing-correction` (number):

```
0.4
```

Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.107 [spacing-interface], page 597, and Section 3.2.113 [staff-spacing-interface], page 601.

3.1.107 StaffSymbol

StaffSymbol objects are created by: Section 2.2.115 [Staff_symbol_engraver], page 347, and Section 2.2.121 [Tab_staff_symbol_engraver], page 350.

Standard settings:

`break-align-symbols` (list):
`'(staff-bar break-alignment)`

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`layer` (integer):
`0`

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-line-thickness` (pair of numbers):
`'(1.0 . 0.1)`

The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`line-count` (integer):
`5`

The number of staff lines.

`stencil` (stencil):
`ly:staff-symbol::print`

The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol::height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.114 [staff-symbol-interface], page 601.

3.1.108 StanzaNumber

StanzaNumber objects are created by: Section 2.2.117 [Stanza_number_engraver], page 348.

Standard settings:

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-series` (symbol):
`'bold`

Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`padding` (dimension, in staff space):
`1.0`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`0`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:text-interface::print`

The symbol to print.

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.104 [side-position-interface], page 593, Section 3.2.116 [stanza-number-interface], page 602, and Section 3.2.125 [text-interface], page 608.

3.1.109 Stem

Stem objects are created by: Section 2.2.111 [Span_stem_engraver], page 347, and Section 2.2.118 [Stem_engraver], page 348.

Standard settings:

`beamlet-default-length` (pair):
`'(1.1 . 1.1)`

A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair):
`'(0.75 . 0.75)`

The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`default-direction` (direction):
`ly:stem::calc-default-direction`

Direction determined by note head positions.

`details` (list):

```
'((lengths 3.5 3.5 3.5 4.25 5.0 6.0)
  (beamed-lengths 3.26 3.5 3.6)
  (beamed-minimum-free-lengths 1.83 1.5 1.25)
  (beamed-extreme-minimum-free-lengths 2.0 1.25)
  (stem-shorten 1.0 0.5 0.25))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:stem::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`double-stem-separation` (number):
`0.5`

The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer):
`stem::calc-duration-log`

The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`length` (dimension, in staff space):
`#<unpure-pure-container #<primitive-procedure ly:stem::calc-length> #<primitive-procedure ly:stem::pure-calc-length> >`

User override for the stem length of unbeamed stems.

`neutral-direction` (direction):
`-1`

Which direction to take in the center of the staff.

`note-collision-threshold` (dimension, in staff space):
`1`

Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`stem-begin-position` (number):
`#<unpure-pure-container #<primitive-procedure ly:stem::calc-stem-begin-position> #<primitive-procedure ly:stem::pure-calc-stem-begin-position> >`

User override for the begin position of a stem.

`stencil` (stencil):
`ly:stem::print`

The symbol to print.

`thickness` (number):
`1.3`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-extent` (pair of numbers):
`ly:stem::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
`ly:stem::offset-callback`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:stem::height> #<primitive-procedure ly:stem::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, and Section 3.2.117 [stem-interface], page 603.

3.1.110 StemStub

StemStub objects are created by: Section 2.2.118 [Stem_engraver], page 348.

Standard settings:

`extra-spacing-height` (pair of numbers):
`stem-stub::extra-spacing-height`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`X-extent` (pair of numbers):
`stem-stub::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #f #<procedure stem-stub::pure-height (grob beg end)> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, and Section 3.2.52 [item-interface], page 568.

3.1.111 StemTremolo

StemTremolo objects are created by: Section 2.2.118 [Stem_engraver], page 348.

Standard settings:

`beam-thickness` (dimension, in staff space):
`0.48`

Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space):
`ly:stem-tremolo::calc-width`

Width of the tremolo sign.

`direction` (direction):
`ly:stem-tremolo::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`shape` (symbol):
`ly:stem-tremolo::calc-shape`

This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

`slope` (number):
`ly:stem-tremolo::calc-slope`

The slope of this object.

`stencil` (stencil):
`ly:stem-tremolo::print`

The symbol to print.

`X-extent` (pair of numbers):
`ly:stem-tremolo::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> #<primitive-procedure ly:stem-tremolo::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:stem-tremolo::calc-y-offset> #<primitive-procedure ly:stem-tremolo::pure-calc-y-offset> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.118 [stem-tremolo-interface], page 605.

3.1.112 StringNumber

StringNumber objects are created by: Section 2.2.75 [New_fingering_engraver], page 335.

Standard settings:

`avoid-slur` (symbol):
`'around`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`font-encoding` (symbol):
`'fetaText`

The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-size` (number):
`-5`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`number-type` (symbol):
`'arabic`

Numbering style. Choices include `roman-lower`, `roman-upper` and `arabic`.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`script-priority` (number):
`100`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number):
`0`

Like `self-alignment-X` but for the Y axis.

`staff-padding` (dimension, in staff space):
`0.5`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`print-circled-text-callback`

The symbol to print.

`text` (markup):
`string-number::calc-text`

Text markup. See Section "Formatting text" in *Notation Reference*.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.80

[number-interface], page 581, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.119 [string-number-interface], page 606, Section 3.2.125 [text-interface], page 608, and Section 3.2.126 [text-script-interface], page 609.

3.1.113 StrokeFinger

StrokeFinger objects are created by: Section 2.2.75 [New_fingering_engraver], page 335.

Standard settings:

`digit-names` (vector):
`#("p" "i" "m" "a" "x")`

Names for string finger digits.

`font-shape` (symbol):
`'italic`

Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number):
`-4`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number):
`0`

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`script-priority` (number):
`100`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number):
`0`

Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number):
`0`

Like `self-alignment-X` but for the Y axis.

`staff-padding` (dimension, in staff space):
`0.5`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`text` (markup):
`stroke-finger::calc-text`
Text markup. See Section "Formatting text" in *Notation Reference*.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.120 [stroke-finger-interface], page 606, Section 3.2.125 [text-interface], page 608, and Section 3.2.126 [text-script-interface], page 609.

3.1.114 SustainPedal

SustainPedal objects are created by: Section 2.2.90 [Piano_pedal_engraver], page 340.

Standard settings:

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`
In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`padding` (dimension, in staff space):
`0.0`
Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)
Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:sustain-pedal::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`
Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.90 [piano-pedal-interface], page 586, Section 3.2.91 [piano-pedal-script-interface], page 586, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.125 [text-interface], page 608.

3.1.115 SustainPedalLineSpanner

SustainPedalLineSpanner objects are created by: Section 2.2.89 [Piano_pedal_align_engraver], page 340.

Standard settings:

`axes` (list):
`'(1)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`-1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`minimum-space` (dimension, in staff space):
`1.0`
Minimum distance that the victim should move (after padding).

`outside-staff-priority` (number):
`1000`
If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`1.2`
Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`1.2`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-element-stencils> #<primitive-procedure ly:grob::pure-vertical-skylines-from-element-stencils> >`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.90 [piano-pedal-interface], page 586, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.116 System

System objects are not created by any engraver.

Standard settings:

`axes` (list):
`'(0 1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`outside-staff-placement-directive` (symbol):
`'left-to-right-polite`

One of four directives telling how outside staff objects should be placed.

- `left-to-right-greedy` – Place each successive grob from left to right.

- `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
- `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
- `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`skyline-horizontal-padding` (number):
`1.0`

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`vertical-skylines` (pair of skylines):
`ly:axis-group-interface::calc-skylines`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:system::height> #<primitive-procedure ly:system::calc-pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.83 [outside-staff-axis-group-interface], page 582, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.121 [system-interface], page 606.

3.1.117 SystemStartBar

SystemStartBar objects are created by: Section 2.2.119 [System_start_delimiter_engraver], page 348.

Standard settings:

`collapse-height` (dimension, in staff space):
`5.0`

Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space):
`-0.1`
Add this much extra space between objects that are next to each other.

`stencil` (stencil):
`ly:system-start-delimiter::print`
The symbol to print.

`style` (symbol):
`'bar-line`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number):
`1.6`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`
The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.122 [system-start-delimiter-interface], page 607.

3.1.118 SystemStartBrace

SystemStartBrace objects are created by: Section 2.2.119 [System_start_delimiter_engraver], page 348.

Standard settings:

`collapse-height` (dimension, in staff space):
`5.0`
Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`direction` (direction):
`-1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-encoding` (symbol):
`'fetaBraces`
The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`padding` (dimension, in staff space):
`0.3`
Add this much extra space between objects that are next to each other.

`stencil` (stencil):
`ly:system-start-delimiter::print`
The symbol to print.

`style` (symbol):
`'brace`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`
The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.122 [system-start-delimiter-interface], page 607.

3.1.119 SystemStartBracket

SystemStartBracket objects are created by: Section 2.2.119 [System_start_delimiter_engraver], page 348.

Standard settings:

`collapse-height` (dimension, in staff space):
`5.0`
Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`direction` (direction):
`-1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space):
`0.8`
Add this much extra space between objects that are next to each other.

`stencil` (stencil):
`ly:system-start-delimiter::print`
The symbol to print.

`style` (symbol):
`'bracket`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number):
`0.45`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

X-offset (number):
ly:side-position-interface::x-aligned-side
The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.122 [system-start-delimiter-interface], page 607.

3.1.120 SystemStartSquare

SystemStartSquare objects are created by: Section 2.2.119 [System_start_delimiter_engraver], page 348.

Standard settings:

collapse-height (dimension, in staff space):
5.0
Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

direction (direction):
-1
If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

stencil (stencil):
ly:system-start-delimiter::print
The symbol to print.

style (symbol):
'line-bracket
This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

thickness (number):
1.0
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

X-offset (number):
ly:side-position-interface::x-aligned-side
The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.122 [system-start-delimiter-interface], page 607.

3.1.121 TabNoteHead

TabNoteHead objects are created by: Section 2.2.120 [Tab_note_heads_engraver], page 349.

Standard settings:

`details` (list):

```
'((cautionary-properties
    (angularity . 0.4)
    (half-thickness . 0.075)
    (padding . 0)
    (procedure
      .
      #<procedure parenthesize-stencil (stencil half-thickness width angulari
    (width . 0.25))
  (head-offset . 3/5)
  (harmonic-properties
    (angularity . 2)
    (half-thickness . 0.075)
    (padding . 0)
    (procedure
      .
      #<procedure parenthesize-stencil (stencil half-thickness width angulari
    (width . 0.25))
  (repeat-tied-properties
    (note-head-visible . #t)
    (parenthesize . #t))
  (tied-properties
    (break-visibility . #(#f #f #t))
    (parenthesize . #t)))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):

`0`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`duration-log` (integer):

`note-head::calc-duration-log`

The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`font-series` (symbol):

`'bold`

Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-size` (number):
`-2`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`parenthesis-friends` (list):
`'(dot)`

A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`stem-attachment` (pair of numbers):
`'(0.0 . 1.35)`

An `(x . y)` pair where the stem attaches to the notehead.

`stencil` (stencil):
`tab-note-head::print`

The symbol to print.

`whiteout` (boolean-or-number):
`#t`

If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`. Usually `#f` by default.

`X-offset` (number):
`ly:self-alignment-interface::x-aligned-on-self`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.77 [note-head-interface], page 580, Section 3.2.96 [rhythmic-grob-interface], page 588, Section 3.2.97 [rhythmic-head-interface], page 588, Section 3.2.115 [staff-symbol-referencer-interface], page 602, Section 3.2.124 [tab-note-head-interface], page 608, and Section 3.2.125 [text-interface], page 608.

3.1.122 TextScript

TextScript objects are created by: Section 2.2.124 [Text_engraver], page 350.

Standard settings:

`avoid-slur` (symbol):
`'around`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`outside-staff-horizontal-padding` (number):
`0.2`

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-priority` (number):
`450`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.3`

Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)

Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`script-priority` (number):
`200`

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`side-axis` (number):
: `1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`slur-padding` (number):
: `0.5`

Extra distance between slur and script.

`staff-padding` (dimension, in staff space):
: `0.5`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
: `ly:text-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
: `#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`X-align-on-main-noteheads` (boolean):
: `#t`

If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-offset` (number):
: `ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
: `#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.51 [instrument-specific-markup-interface],

page 566, Section 3.2.52 [item-interface], page 568, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.100 [self-alignment-interface], page 590, Section 3.2.104 [side-position-interface], page 593, Section 3.2.125 [text-interface], page 608, and Section 3.2.126 [text-script-interface], page 609.

3.1.123 TextSpanner

TextSpanner objects are created by: Section 2.2.125 [Text_spanner_engraver], page 351.

Standard settings:

`bound-details` (list):

```
'((left (Y . 0) (padding . 0.25) (attach-dir . -1))
  (left-broken (attach-dir . 1))
  (right (Y . 0) (padding . 0.25)))
```

An alist of properties for determining attachments of spanners to edges.

`dash-fraction` (number):

`0.2`

Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number):

`3.0`

The length of one dash together with whitespace. If negative, no line is drawn at all.

`direction` (direction):

`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-shape` (symbol):

`'italic`

Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`left-bound-info` (list):

`ly:line-spanner::calc-left-bound-info`

An alist of properties for determining attachments of spanners to edges.

`outside-staff-priority` (number):

`350`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`right-bound-info` (list):

`ly:line-spanner::calc-right-bound-info`

An alist of properties for determining attachments of spanners to edges.

`side-axis` (number):

`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`0.8`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:line-spanner::print`

The symbol to print.

`style` (symbol):
`'dashed-line`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.124 Tie

Tie objects are created by: Section 2.2.20 [Completion_heads_engraver], page 315, and Section 2.2.126 [Tie_engraver], page 351.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`control-points` (list of number pairs):
`ly:tie::calc-control-points`

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list):
```
'((ratio . 0.333)
  (center-staff-line-clearance . 0.6)
  (tip-staff-line-clearance . 0.45)
  (note-head-gap . 0.2)
  (stem-gap . 0.35)
```

```
(height-limit . 1.0)
(horizontal-distance-penalty-factor . 10)
(same-dir-as-stem-penalty . 8)
(min-length-penalty-factor . 26)
(tie-tie-collision-distance . 0.45)
(tie-tie-collision-penalty . 25.0)
(intra-space-threshold . 1.25)
(outer-tie-vertical-distance-symmetry-penalty-factor
  .
  10)
(outer-tie-length-symmetry-penalty-factor . 10)
(vertical-distance-penalty-factor . 7)
(outer-tie-vertical-gap . 0.25)
(multi-tie-region-size . 3)
(single-tie-region-size . 4)
(between-length-limit . 1.0))
```

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction):
`ly:tie::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-size` (number):
`-6`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`line-thickness` (number):
`0.8`

For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`neutral-direction` (direction):
`1`

Which direction to take in the center of the staff.

`springs-and-rods` (boolean):
`ly:spanner::set-spacing-rods`

Dummy variable for triggering spacing routines.

`stencil` (stencil):
`ly:tie::print`

The symbol to print.

`thickness` (number):
`1.2`

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`

Two skylines, one above and one below this grob.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.128 [tie-interface], page 610.

3.1.125 TieColumn

TieColumn objects are created by: Section 2.2.20 [Completion_heads_engraver], page 315, and Section 2.2.126 [Tie_engraver], page 351.

Standard settings:

`before-line-breaking` (boolean):
`ly:tie-column::before-line-breaking`

Dummy property, used to trigger a callback function.

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.127 [tie-column-interface], page 609.

3.1.126 TimeSignature

TimeSignature objects are created by: Section 2.2.128 [Time_signature_engraver], page 352.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`break-align-anchor` (number):
`ly:break-aligned-interface::calc-extent-aligned-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number):
`-1`

Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-symbol` (symbol):
`'time-signature`

This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-visibility` (vector):
`#(#t #t #t)`

A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):
`pure-from-neighbor-interface::extra-spacing-height-including-staff`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):
`'(0.0 . 0.8)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`space-alist` (list):
```
'((cue-clef extra-space . 1.5)
  (first-note fixed-space . 2.0)
  (right-edge extra-space . 0.5)
  (staff-bar extra-space . 1.0))
```

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special

break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil` (stencil):
: `ly:time-signature::print`

The symbol to print.

`style` (symbol):
: `'C`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.15 [break-aligned-interface], page 545, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.93 [pure-from-neighbor-interface], page 587, and Section 3.2.129 [time-signature-interface], page 613.

3.1.127 TrillPitchAccidental

TrillPitchAccidental objects are created by: Section 2.2.93 [Pitched_trill_engraver], page 341.

Standard settings:

`direction` (direction):
`-1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-size` (number):
`-4`
The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`glyph-name-alist` (list):
```
'((0 . "accidentals.natural")
  (-1/2 . "accidentals.flat")
  (1/2 . "accidentals.sharp")
  (1 . "accidentals.doublesharp")
  (-1 . "accidentals.flatflat")
  (3/4
   .
   "accidentals.sharp.slashslash.stemstemstem")
  (1/4 . "accidentals.sharp.slashslash.stem")
  (-1/4 . "accidentals.mirroredflat")
  (-3/4 . "accidentals.mirroredflat.flat"))
```
An alist of key-string pairs.

`padding` (dimension, in staff space):
`0.2`
Add this much extra space between objects that are next to each other.

`side-axis` (number):
`0`
If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`ly:accidental-interface::print`
The symbol to print.

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`
The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:accidental-interface::height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.1 [accidental-interface], page 534, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.50 [inline-accidental-interface], page 566, Section 3.2.52 [item-interface], page 568, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.130 [trill-pitch-accidental-interface], page 613.

3.1.128 TrillPitchGroup

TrillPitchGroup objects are created by: Section 2.2.93 [Pitched_trill_engraver], page 341.

Standard settings:

`axes` (list):
`'(0)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-size` (number):
`-4`
The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`horizon-padding` (number):
`0.1`
The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`minimum-space` (dimension, in staff space):
`2.5`
Minimum distance that the victim should move (after padding).

`padding` (dimension, in staff space):
`0.3`
Add this much extra space between objects that are next to each other.

`side-axis` (number):
`0`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`stencil` (stencil):
`parenthesize-elements`

The symbol to print.

`stencils` (list):
`parentheses-item::calc-parenthesis-stencils`

Multiple stencils, used as intermediate value.

`X-offset` (number):
`ly:side-position-interface::x-aligned-side`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.77 [note-head-interface], page 580, Section 3.2.86 [parentheses-interface], page 585, and Section 3.2.104 [side-position-interface], page 593.

3.1.129 TrillPitchHead

TrillPitchHead objects are created by: Section 2.2.93 [Pitched_trill_engraver], page 341.

Standard settings:

`duration-log` (integer):
`2`

The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`font-size` (number):
`-4`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`stencil` (stencil):
`ly:note-head::print`

The symbol to print.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:staff-symbol-referencer::callback> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.57 [ledgered-interface], page 571, Section 3.2.92 [pitched-trill-interface], page 586, Section 3.2.97 [rhythmic-head-interface], page 588, and Section 3.2.115 [staff-symbol-referencer-interface], page 602.

3.1.130 TrillSpanner

TrillSpanner objects are created by: Section 2.2.131 [Trill_spanner_engraver], page 353.

Standard settings:

`after-line-breaking` (boolean):
`ly:spanner::kill-zero-spanned-time`

Dummy property, used to trigger callback for `after-line-breaking`.

`bound-details` (list):

```
'((left (text #<procedure musicglyph-markup (layout props glyph-name)>
                "scripts.trill")
        (Y . 0)
        (stencil-offset -0.5 . -1)
        (padding . 0.5)
        (attach-dir . 0))
  (left-broken (end-on-note . #t))
  (right (Y . 0)))
```

An alist of properties for determining attachments of spanners to edges.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`left-bound-info` (list):
`ly:line-spanner::calc-left-bound-info`

An alist of properties for determining attachments of spanners to edges.

`outside-staff-priority` (number):
`50`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`0.5`

Add this much extra space between objects that are next to each other.

`right-bound-info` (list):
`ly:line-spanner::calc-right-bound-info`

An alist of properties for determining attachments of spanners to edges.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`1.0`

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:line-spanner::print`

The symbol to print.

`style` (symbol):
`'trill`

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.131 [trill-spanner-interface], page 613.

3.1.131 TupletBracket

TupletBracket objects are created by: Section 2.2.132 [Tuplet_engraver], page 353.

Standard settings:

`avoid-scripts` (boolean):
`#t`

If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`connect-to-neighbor` (pair):
`ly:tuplet-bracket::calc-connect-to-neighbors`

Pair of booleans, indicating whether this grob looks as a continued break.

`direction` (direction):
`ly:tuplet-bracket::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair):
`'(0.7 . 0.7)`
A pair of numbers specifying the heights of the vertical edges: `(left-height . right-height)`.

`full-length-to-extent` (boolean):
`#t`
Run to the extent of the column for a full-length tuplet bracket.

`padding` (dimension, in staff space):
`1.1`
Add this much extra space between objects that are next to each other.

`positions` (pair of numbers):
`ly:tuplet-bracket::calc-positions`
Pair of staff coordinates `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`shorten-pair` (pair of numbers):
`'(-0.2 . -0.2)`
The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`staff-padding` (dimension, in staff space):
`0.25`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`stencil` (stencil):
`ly:tuplet-bracket::print`
The symbol to print.

`thickness` (number):
`1.6`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`X-positions` (pair of numbers):
`ly:tuplet-bracket::calc-x-positions`
Pair of X staff coordinates of a spanner in the form `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.132 [tuplet-bracket-interface], page 613.

3.1.132 TupletNumber

TupletNumber objects are created by: Section 2.2.132 [Tuplet_engraver], page 353.

Standard settings:

`avoid-slur` (symbol):
`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`direction` (direction):
`tuplet-number::calc-direction`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`font-shape` (symbol):
`'italic`

Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number):
`-2`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`knee-to-beam` (boolean):
`#t`

Determines whether a tuplet number will be positioned next to a kneed beam.

`stencil` (stencil):
`ly:tuplet-number::print`

The symbol to print.

`text` (markup):
`tuplet-number::calc-denominator-text`

Text markup. See Section "Formatting text" in *Notation Reference*.

`X-offset` (number):
`ly:tuplet-number::calc-x-offset`

The horizontal amount that this object is moved relative to its X-parent.

`Y-offset` (number):
`ly:tuplet-number::calc-y-offset`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.111 [spanner-interface], page 599, Section 3.2.125 [text-interface], page 608, and Section 3.2.133 [tuplet-number-interface], page 615.

3.1.133 UnaCordaPedal

UnaCordaPedal objects are created by: Section 2.2.90 [Piano_pedal_engraver], page 340.

Standard settings:

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`extra-spacing-width` (pair of numbers):
`'(+inf.0 . -inf.0)`
In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`font-shape` (symbol):
`'italic`
Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`padding` (dimension, in staff space):
`0.0`
Add this much extra space between objects that are next to each other.

`parent-alignment-X` (number)
Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):
`0`
Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`stencil` (stencil):
`ly:text-interface::print`
The symbol to print.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> >`

Two skylines, one above and one below this grob.

`X-offset` (number):
`ly:self-alignment-interface::aligned-on-x-parent`

The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.52 [item-interface], page 568, Section 3.2.91 [piano-pedal-script-interface], page 586, Section 3.2.100 [self-alignment-interface], page 590, and Section 3.2.125 [text-interface], page 608.

3.1.134 UnaCordaPedalLineSpanner

UnaCordaPedalLineSpanner objects are created by: Section 2.2.89 [Piano_pedal_align_engraver], page 340.

Standard settings:

`axes` (list):
`'(1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`-1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`minimum-space` (dimension, in staff space):
`1.0`

Minimum distance that the victim should move (after padding).

`outside-staff-priority` (number):
`1000`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`1.2`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`staff-padding` (dimension, in staff space):
`1.2`
Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-element-stencils> #<primitive-procedure ly:grob::pure-vertical-skylines-from-element-stencils> >`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.90 [piano-pedal-interface], page 586, Section 3.2.104 [side-position-interface], page 593, and Section 3.2.111 [spanner-interface], page 599.

3.1.135 VaticanaLigature

VaticanaLigature objects are created by: Section 2.2.134 [Vaticana_ligature_engraver], page 354.

Standard settings:

`stencil` (stencil):
`ly:vaticana-ligature::print`
The symbol to print.

`thickness` (number):
`0.6`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.135 [vaticana-ligature-interface], page 616.

3.1.136 VerticalAlignment

VerticalAlignment objects are created by: Section 2.2.135 [Vertical_align_engraver], page 354.

Standard settings:

`axes` (list):
`'(1)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`stacking-dir` (direction):
`-1`
Stack objects in which direction?

`vertical-skylines` (pair of skylines):
`ly:axis-group-interface::combine-skylines`
Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.4 [align-interface], page 535, Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, and Section 3.2.111 [spanner-interface], page 599.

3.1.137 VerticalAxisGroup

VerticalAxisGroup objects are created by: Section 2.2.5 [Axis_group_engraver], page 309.

Standard settings:

`axes` (list):
`'(1)`
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`default-staff-staff-spacing` (list):
```
'((basic-distance . 9)
  (minimum-distance . 8)
  (padding . 1))
```
The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`nonstaff-unrelatedstaff-spacing` (list):
`'((padding . 0.5))`
The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from

`staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`outside-staff-placement-directive` (symbol):
`'left-to-right-polite`

One of four directives telling how outside staff objects should be placed.

- `left-to-right-greedy` – Place each successive grob from left to right.
- `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
- `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
- `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`skyline-horizontal-padding` (number):
`0.1`

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`staff-staff-spacing` (list):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::calc-staff-staff-spacing> #<primitive-procedure ly:axis-group-interface::calc-pure-staff-staff-spacing> >`

When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

- `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`stencil` (stencil):
`ly:axis-group-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):
`ly:hara-kiri-group-spanner::calc-skylines`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:hara-kiri-group-spanner::y-extent> #<primitive-procedure ly:hara-kiri-group-spanner::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
`ly:hara-kiri-group-spanner::force-hara-kiri-callback`

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.47 [hara-kiri-group-spanner-interface], page 564, Section 3.2.83 [outside-staff-axis-group-interface], page 582, and Section 3.2.111 [spanner-interface], page 599.

3.1.138 VoiceFollower

VoiceFollower objects are created by: Section 2.2.76 [Note_head_line_engraver], page 336.

Standard settings:

`after-line-breaking` (boolean):
`ly:spanner::kill-zero-spanned-time`

Dummy property, used to trigger callback for `after-line-breaking`.

`bound-details` (list):
```
'((right (attach-dir . 0) (padding . 1.5))
  (left (attach-dir . 0) (padding . 1.5)))
```

An alist of properties for determining attachments of spanners to edges.

`gap` (dimension, in staff space):
`0.5`

Size of a gap in a variable symbol.

`left-bound-info` (list):
`ly:line-spanner::calc-left-bound-info`

An alist of properties for determining attachments of spanners to edges.

`non-musical` (boolean):
`#t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`right-bound-info` (list):
`ly:line-spanner::calc-right-bound-info`
An alist of properties for determining attachments of spanners to edges.

`stencil` (stencil):
`ly:line-spanner::print`
The symbol to print.

`style` (symbol):
`'line`
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`X-extent` (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.45 [grob-interface], page 559, Section 3.2.61 [line-interface], page 572, Section 3.2.62 [line-spanner-interface], page 573, and Section 3.2.111 [spanner-interface], page 599.

3.1.139 VoltaBracket

VoltaBracket objects are created by: Section 2.2.136 [Volta_engraver], page 355.

Standard settings:

`baseline-skip` (dimension, in staff space):
`1.7`
Distance between base lines of multiple lines of text.

`direction` (direction):
`1`
If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair):
`'(2.0 . 2.0)`
A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`font-encoding` (symbol):
`'fetaText`
The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-size` (number):
`-4`

The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`shorten-pair` (pair of numbers):
`ly:volta-bracket::calc-shorten-pair`
The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`stencil` (stencil):
`ly:volta-bracket-interface::print`
The symbol to print.

`thickness` (number):
`1.6`
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-stencil> #<primitive-procedure ly:grob::pure-simple-vertical-skylines-from-extents> >`
Two skylines, one above and one below this grob.

`word-space` (dimension, in staff space):
`0.6`
Space to insert between words in texts.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:grob::stencil-height> #<procedure volta-bracket-interface::pure-height (grob start end)> >`
Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): Section 3.2.36 [font-interface], page 553, Section 3.2.45 [grob-interface], page 559, Section 3.2.48 [horizontal-bracket-interface], page 565, Section 3.2.61 [line-interface], page 572, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, Section 3.2.125 [text-interface], page 608, Section 3.2.136 [volta-bracket-interface], page 617, and Section 3.2.137 [volta-interface], page 617.

3.1.140 VoltaBracketSpanner

VoltaBracketSpanner objects are created by: Section 2.2.136 [Volta_engraver], page 355.

Standard settings:

`after-line-breaking` (boolean):
`ly:side-position-interface::move-to-extremal-staff`

Dummy property, used to trigger callback for `after-line-breaking`.

`axes` (list):
`'(1)`

List of axis numbers. In the case of alignment grobs, this should contain only one number.

`direction` (direction):
`1`

If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`no-alignment` (boolean):
`#t`

If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`outside-staff-priority` (number):
`600`

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`padding` (dimension, in staff space):
`1`

Add this much extra space between objects that are next to each other.

`side-axis` (number):
`1`

If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`vertical-skylines` (pair of skylines):
`#<unpure-pure-container #<primitive-procedure ly:grob::vertical-skylines-from-element-stencils> #<primitive-procedure ly:grob::pure-vertical-skylines-from-element-stencils> >`

Two skylines, one above and one below this grob.

`X-extent` (pair of numbers):
`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`Y-extent` (pair of numbers):
`#<unpure-pure-container #<primitive-procedure ly:axis-group-interface::height> #<primitive-procedure ly:axis-group-interface::pure-height> >`

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number):
: `#<unpure-pure-container #<primitive-procedure ly:side-position-interface::y-aligned-side> #<primitive-procedure ly:side-position-interface::pure-y-aligned-side> >`

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): Section 3.2.7 [axis-group-interface], page 537, Section 3.2.45 [grob-interface], page 559, Section 3.2.84 [outside-staff-interface], page 582, Section 3.2.104 [side-position-interface], page 593, Section 3.2.111 [spanner-interface], page 599, and Section 3.2.137 [volta-interface], page 617.

3.2 Graphical Object Interfaces

3.2.1 accidental-interface

A single accidental.

User settable properties:

`alteration` (number)
: Alteration numbers for accidental.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`parenthesized` (boolean)
: Parenthesize this grob.

`restore-first` (boolean)
: Print a natural before the accidental.

Internal properties:

`forced` (boolean)
: Manually forced accidental.

`tie` (graphical (layout) object)
: A pointer to a `Tie` object.

This grob interface is used in the following graphical object(s): Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.6 [AmbitusAccidental], page 376, and Section 3.1.127 [TrillPitchAccidental], page 518.

3.2.2 accidental-placement-interface

Resolve accidental collisions.

User settable properties:

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

Internal properties:

`accidental-grobs` (list)
: An alist with `(`*`notename . groblist`*`)` entries.

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.3 [AccidentalPlacement], page 372.

3.2.3 accidental-suggestion-interface

An accidental, printed as a suggestion (typically: vertically over a note).

This grob interface is used in the following graphical object(s): Section 3.1.4 [AccidentalSuggestion], page 373.

3.2.4 align-interface

Order grobs from top to bottom, left to right, right to left or bottom to top. For vertical alignments of staves, the `line-break-system-details` of the left Section "NonMusicalPaperColumn" in *Internals Reference* may be set to tune vertical spacing.

User settable properties:

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`axes` (list) List of axis numbers. In the case of alignment grobs, this should contain only one number.

`padding` (dimension, in staff space)
Add this much extra space between objects that are next to each other.

`stacking-dir` (direction)
Stack objects in which direction?

Internal properties:

`elements` (array of grobs)
An array of grobs; the type is depending on the grob where this is set in.

`minimum-translations-alist` (list)
An list of translations for a given start and end point.

`positioning-done` (boolean)
Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.14 [BassFigureAlignment], page 387, and Section 3.1.136 [VerticalAlignment], page 528.

3.2.5 ambitus-interface

The line between note heads for a pitch range.

User settable properties:

`gap` (dimension, in staff space)
Size of a gap in a variable symbol.

`length-fraction` (number)
Multiplier for lengths. Used for determining ledger lines and stem lengths.

`maximum-gap` (number)
Maximum value allowed for `gap` property.

`thickness` (number)
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`note-heads` (array of grobs)
An array of note head grobs.

This grob interface is used in the following graphical object(s): Section 3.1.5 [Ambitus], page 375, Section 3.1.7 [AmbitusLine], page 377, and Section 3.1.8 [AmbitusNoteHead], page 378.

3.2.6 arpeggio-interface

Functions and settings for drawing an arpeggio symbol.

User settable properties:

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`positions` (pair of numbers)
: Pair of staff coordinates (*`left . right`*), where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`protrusion` (number)
: In an arpeggio bracket, the length of the horizontal edges.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

Internal properties:

`stems` (array of grobs)
: An array of stem objects.

This grob interface is used in the following graphical object(s): Section 3.1.9 [Arpeggio], page 379.

3.2.7 axis-group-interface

An object that groups other layout objects.

User settable properties:

`axes` (list) List of axis numbers. In the case of alignment grobs, this should contain only one number.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the

relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.

- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

Internal properties:

`adjacent-pure-heights` (pair)
: A pair of vectors. Used by a `VerticalAxisGroup` to cache the `Y-extent`s of different column ranges.

`bound-alignment-interfaces` (list)
: Interfaces to be used for positioning elements that align with a column.

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

`pure-relevant-grobs` (array of grobs)
: All the grobs (items and spanners) that are relevant for finding the `pure-Y-extent`

`pure-relevant-items` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-relevant-spanners` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-Y-common` (graphical (layout) object)
: A cache of the `common_refpoint_of_array` of the `elements` grob set.

`staff-grouper` (graphical (layout) object)
: The staff grouper we belong to.

`system-Y-offset` (number)
: The Y-offset (relative to the bottom of the top-margin of the page) of the system to which this staff belongs.

`X-common` (graphical (layout) object)
: Common reference point for axis group.

`Y-common` (graphical (layout) object)
: See `X-common`.

This grob interface is used in the following graphical object(s): Section 3.1.5 [Ambitus], page 375, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, Section 3.1.33 [DotColumn], page 412, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.78 [NoteCollision], page 465, Section 3.1.79 [NoteColumn], page 466, Section 3.1.84 [PaperColumn], page 470, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.116 [System], page 503, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, Section 3.1.136 [VerticalAlignment], page 528, Section 3.1.137 [VerticalAxisGroup], page 528, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.2.8 balloon-interface

A collection of routines to put text balloons around an object.

User settable properties:

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

Internal properties:

`spanner-placement` (direction)
: The place of an annotation on a spanner. `LEFT` is for the first spanner, and `RIGHT` is for the last. `CENTER` will place it on the broken spanner that falls closest to the center of the length of the entire spanner, although this behavior is unpredictable in situations with lots of rhythmic diversity. For predictable results, use `LEFT` and `RIGHT`.

This grob interface is used in the following graphical object(s): Section 3.1.10 [BalloonTextItem], page 381, Section 3.1.45 [FootnoteItem], page 426, and Section 3.1.46 [FootnoteSpanner], page 427.

3.2.9 bar-line-interface

Print a special bar symbol. It replaces the regular bar symbol with a special symbol. The argument *bartype* is a string which specifies the kind of bar line to print.

The list of allowed glyphs and predefined bar lines can be found in `scm/bar-line.scm`.

`gap` is used for the gaps in dashed bar lines.

User settable properties:

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`bar-extent` (pair of numbers)
: The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`hair-thickness` (number)
: Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`kern` (dimension, in staff space)
: The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`rounded` (boolean)
: Decide whether lines should be drawn rounded or not.

`segno-kern` (number)
: The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`thick-thickness` (number)
: Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

Internal properties:

`has-span-bar` (pair)
: A pair of grobs containing the span bars to be drawn below and above the staff. If no span bar is in a position, the respective element is set to `#f`.

This grob interface is used in the following graphical object(s): Section 3.1.11 [BarLine], page 381, and Section 3.1.103 [SpanBar], page 490.

3.2.10 bass-figure-alignment-interface

Align a bass figure.

This grob interface is used in the following graphical object(s): Section 3.1.14 [BassFigureAlignment], page 387.

3.2.11 bass-figure-interface

A bass figure text.

User settable properties:

`implicit` (boolean)
: Is this an implicit bass figure?

This grob interface is used in the following graphical object(s): Section 3.1.13 [BassFigure], page 386.

3.2.12 beam-interface

A beam.

The `beam-thickness` property is the weight of beams, measured in staffspace. The `direction` property is not user-serviceable. Use the `direction` property of `Stem` instead. The following properties may be set in the `details` list.

`stem-length-demerit-factor`
: Demerit factor used for inappropriate stem lengths.

`secondary-beam-demerit`
: Demerit used in quanting calculations for multiple beams.

`region-size`
: Size of region for checking quant scores.

`beam-eps` Epsilon for beam quant code to check for presence in gap.

`stem-length-limit-penalty`
: Penalty for differences in stem lengths on a beam.

`damping-direction-penalty`
: Demerit penalty applied when beam direction is different from damping direction.

`hint-direction-penalty`
: Demerit penalty applied when beam direction is different from damping direction, but damping slope is <= `round-to-zero-slope`.

`musical-direction-factor`
: Demerit scaling factor for difference between beam slope and music slope.

`ideal-slope-factor`
: Demerit scaling factor for difference between beam slope and damping slope.

`round-to-zero-slope`
: Damping slope which is considered zero for purposes of calculating direction penalties.

User settable properties:

`annotation` (string)
: Annotate a grob for debug purposes.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beamed-stem-shorten` (list)
: How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`damping` (number)
: Amount of beam slope damping.

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`grow-direction` (direction)
: Crescendo or decrescendo?

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`knee` (boolean)
: Is this beam kneed?

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`positions` (pair of numbers)
: Pair of staff coordinates `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`skip-quanting` (boolean)
: Should beam quanting be skipped?

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff.

Internal properties:

`beam-segments` (list)
: Internal representation of beam segments.

`covered-grobs` (array of grobs)
: Grobs that could potentially collide with a beam.

`least-squares-dy` (number)
: The ideal beam slope, without damping.

`normal-stems` (array of grobs)
: An array of visible stems.

`quantized-positions` (pair of numbers)
: The beam positions after quanting.

`shorten` (dimension, in staff space)
: The amount of space that a stem is shortened. Internally used to distribute beam shortening over stems.

`stems` (array of grobs)
: An array of stem objects.

This grob interface is used in the following graphical object(s): Section 3.1.19 [Beam], page 390.

3.2.13 bend-after-interface

A doit or drop.

User settable properties:

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`delta-position` (number)
: The vertical position difference.

This grob interface is used in the following graphical object(s): Section 3.1.20 [BendAfter], page 392.

3.2.14 break-alignable-interface

Object that is aligned on a break alignment.

User settable properties:

`break-align-symbols` (list)
: A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

This grob interface is used in the following graphical object(s): Section 3.1.12 [BarNumber], page 385, Section 3.1.73 [MetronomeMark], page 458, and Section 3.1.90 [RehearsalMark], page 477.

3.2.15 break-aligned-interface

Breakable items.

User settable properties:

`break-align-anchor` (number)
: Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-symbol` (symbol)
: This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`space-alist` (list)
: An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *spacing-style* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

This grob interface is used in the following graphical object(s): Section 3.1.5 [Ambitus], page 375, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.11 [BarLine], page 381, Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.23 [BreathingSign], page 394, Section 3.1.25 [Clef], page 397, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.63 [LeftEdge], page 447, and Section 3.1.126 [TimeSignature], page 515.

3.2.16 break-alignment-interface

The object that performs break alignment.

Three interfaces deal specifically with break alignment:

1. break-alignment-interface (this one),
2. Section 3.2.14 [break-alignable-interface], page 544, and
3. Section 3.2.15 [break-aligned-interface], page 545.

Each of these interfaces supports grob properties that use *break-align symbols*, which are Scheme symbols that are used to specify the alignment, ordering, and spacing of certain notational elements (‘breakable’ items).

Available break-align symbols:

ambitus
breathing-sign
clef
cue-clef
cue-end-clef
custos
key-cancellation
key-signature
left-edge
staff-bar
time-signature

User settable properties:

`break-align-orders` (vector)
: This is a vector of 3 lists: `#(end-of-line unbroken start-of-line)`. Each list contains *break-align symbols* that specify an order of breakable items (see Section "break-alignment-interface" in *Internals Reference*).

  For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
                    ambitus
                    breathing-sign
                    time-signature
                    clef
                    cue-clef
                    staff-bar
                    key-cancellation
                    key-signature
                    custos))
```

Internal properties:

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.22 [BreakAlignment], page 393.

3.2.17 breathing-sign-interface

A breathing sign.

User settable properties:

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

This grob interface is used in the following graphical object(s): Section 3.1.23 [BreathingSign], page 394.

3.2.18 chord-name-interface

A chord label (name or fretboard).

Internal properties:

`begin-of-line-visible` (boolean)
: Set to make `ChordName` or `FretBoard` be visible only at beginning of line or at chord changes.

This grob interface is used in the following graphical object(s): Section 3.1.24 [ChordName], page 396, and Section 3.1.47 [FretBoard], page 428.

3.2.19 clef-interface

A clef sign.

User settable properties:

`full-size-change` (boolean)
: Don't make a change clef smaller.

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`non-default` (boolean)
: Set for manually specified clefs and keys.

This grob interface is used in the following graphical object(s): Section 3.1.25 [Clef], page 397, Section 3.1.30 [CueClef], page 404, and Section 3.1.31 [CueEndClef], page 407.

3.2.20 clef-modifier-interface

The number describing transposition of the clef, placed below or above clef sign. Usually this is 8 (octave transposition) or 15 (two octaves), but LilyPond allows any integer here.

User settable properties:

`clef-alignments` (list)
: An alist of parent-alignments that should be used for clef modifiers with various clefs

This grob interface is used in the following graphical object(s): Section 3.1.26 [ClefModifier], page 400.

3.2.21 cluster-beacon-interface

A place holder for the cluster spanner to determine the vertical extents of a cluster spanner at this X position.

User settable properties:

`positions` (pair of numbers)
: Pair of staff coordinates `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

This grob interface is used in the following graphical object(s): Section 3.1.28 [ClusterSpannerBeacon], page 402.

3.2.22 cluster-interface

A graphically drawn musical cluster.

`padding` adds to the vertical extent of the shape (top and bottom).

The property `style` controls the shape of cluster segments. Valid values include `leftsided-stairs`, `rightsided-stairs`, `centered-stairs`, and `ramp`.

User settable properties:

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

Internal properties:

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

This grob interface is used in the following graphical object(s): Section 3.1.27 [ClusterSpanner], page 402.

3.2.23 custos-interface

A custos object. `style` can have four valid values: `mensural`, `vaticana`, `medicaea`, and `hufnagel`. `mensural` is the default style.

User settable properties:

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

This grob interface is used in the following graphical object(s): Section 3.1.32 [Custos], page 410.

3.2.24 dot-column-interface

Group dot objects so they form a column, and position dots so they do not clash with staff lines.

User settable properties:

`chord-dots-limit` (integer)
: Limits the column of dots on each chord to the height of the chord plus `chord-dots-limit` staff-positions.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP=1`, `DOWN=-1`, `LEFT=-1`, `RIGHT=1`, `CENTER=0`.

Internal properties:

`dots` (array of grobs)
: Multiple `Dots` objects.

`note-collision` (graphical (layout) object)
: The `NoteCollision` object of a dot column.

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.33 [DotColumn], page 412.

3.2.25 dots-interface

The dots to go with a notehead or rest. `direction` sets the preferred direction to move in case of staff line collisions. `style` defaults to undefined, which is normal 19th/20th century traditional style. Set `style` to `vaticana` for ancient type dots.

User settable properties:

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP=1`, `DOWN=-1`, `LEFT=-1`, `RIGHT=1`, `CENTER=0`.

`dot-count` (integer)
: The number of dots.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

This grob interface is used in the following graphical object(s): Section 3.1.34 [Dots], page 412.

3.2.26 dynamic-interface

Any kind of loudness sign.

This grob interface is used in the following graphical object(s): Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, and Section 3.1.52 [Hairpin], page 432.

3.2.27 dynamic-line-spanner-interface

Dynamic line spanner.

User settable properties:

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

This grob interface is used in the following graphical object(s): Section 3.1.38 [DynamicLineSpanner], page 417.

3.2.28 dynamic-text-interface

An absolute text dynamic.

User settable properties:

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

This grob interface is used in the following graphical object(s): Section 3.1.39 [DynamicText], page 418.

3.2.29 dynamic-text-spanner-interface

Dynamic text spanner.

User settable properties:

`text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

This grob interface is used in the following graphical object(s): Section 3.1.40 [DynamicTextSpanner], page 420.

3.2.30 enclosing-bracket-interface

Brackets alongside bass figures.

User settable properties:

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

This grob interface is used in the following graphical object(s): Section 3.1.16 [BassFigureBracket], page 389.

3.2.31 episema-interface

An episema line.

This grob interface is used in the following graphical object(s): Section 3.1.41 [Episema], page 422.

3.2.32 figured-bass-continuation-interface

Simple extender line between bounds.

User settable properties:

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`figures` (array of grobs)
: Figured bass objects for continuation line.

This grob interface is used in the following graphical object(s): Section 3.1.17 [BassFigureContinuation], page 389.

3.2.33 finger-interface

A fingering instruction.

This grob interface is used in the following graphical object(s): Section 3.1.42 [Fingering], page 422.

3.2.34 fingering-column-interface

Makes sure that fingerings placed laterally do not collide and that they are flush if necessary.

User settable properties:

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`snap-radius` (number)
: The maximum distance between two objects that will cause them to snap to alignment along an axis.

Internal properties:

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.43 [FingeringColumn], page 424.

3.2.35 flag-interface

A flag that gets attached to a stem.The style property is symbol determining what style of flag glyph is typeset on a `Stem`. Valid options include `'()` for standard flags, `'mensural` and `'no-flag`, which switches off the flag.

User settable properties:

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

This grob interface is used in the following graphical object(s): Section 3.1.44 [Flag], page 425.

3.2.36 font-interface

Any symbol that is typeset through fixed sets of glyphs, (i.e., fonts).

User settable properties:

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-features` (list)
: Opentype features.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

Internal properties:

`font` (font metric)
: A cached font metric object.

This grob interface is used in the following graphical object(s): Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.7 [AmbitusLine], page 377, Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.9 [Arpeggio], page 379, Section 3.1.10 [BalloonTextItem], page 381, Section 3.1.11 [BarLine], page 381, Section 3.1.12 [BarNumber], page 385, Section 3.1.13 [BassFigure], page 386, Section 3.1.19 [Beam], page 390, Section 3.1.23 [BreathingSign], page 394, Section 3.1.24 [ChordName], page 396, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.42 [Fingering], page 422, Section 3.1.44 [Flag], page 425, Section 3.1.45 [FootnoteItem], page 426, Section 3.1.46 [FootnoteSpanner], page 427, Section 3.1.47 [FretBoard], page 428, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.59 [KievanLigature], page 444, Section 3.1.66 [LyricHyphen], page 451, Section 3.1.68 [LyricText], page 453, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.72 [MensuralLigature], page 457, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.80 [NoteHead], page 467, Section 3.1.81 [NoteName], page 468, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.84 [PaperColumn], page 470, Section 3.1.85 [ParenthesesItem], page 471, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.94 [Rest], page 481, Section 3.1.96 [Script], page 483, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.103 [SpanBar], page 490, Section 3.1.108 [StanzaNumber], page 493, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.114

[SustainPedal], page 501, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, Section 3.1.121 [TabNoteHead], page 508, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.126 [TimeSignature], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.132 [TupletNumber], page 524, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.135 [VaticanaLigature], page 527, and Section 3.1.139 [VoltaBracket], page 531.

3.2.37 footnote-interface

Make a footnote.

User settable properties:

`automatically-numbered` (boolean)
: Should a footnote be automatically numbered?

`footnote` (boolean)
: Should this be a footnote or in-note?

`footnote-text` (markup)
: A footnote for the grob.

Internal properties:

`numbering-assertion-function` (any type)
: The function used to assert that footnotes are receiving correct automatic numbers.

This grob interface is used in the following graphical object(s): Section 3.1.45 [FootnoteItem], page 426, and Section 3.1.46 [FootnoteSpanner], page 427.

3.2.38 footnote-spanner-interface

Make a footnote spanner.

User settable properties:

`footnote-text` (markup)
: A footnote for the grob.

Internal properties:

`spanner-placement` (direction)
: The place of an annotation on a spanner. `LEFT` is for the first spanner, and `RIGHT` is for the last. `CENTER` will place it on the broken spanner that falls closest to the center of the length of the entire spanner, although this behavior is unpredictable in situations with lots of rhythmic diversity. For predictable results, use `LEFT` and `RIGHT`.

This grob interface is used in the following graphical object(s): Section 3.1.46 [FootnoteSpanner], page 427.

3.2.39 fret-diagram-interface

A fret diagram

User settable properties:

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`dot-placement-list` (list)
: List consisting of `(`*`description string-number fret-number finger-number`*`)` entries used to define fret diagrams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a `(`*`property . value`*`)` pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
  - `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
  - `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
  - `paren-padding` – The padding for the parenthesis. Default 0.05.
  - `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
  - `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
  - `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later

case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.

- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`size` (number)
: The ratio of the size of the object to its default size.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.47 [FretBoard], page 428.

3.2.40 glissando-interface

A glissando.

Internal properties:

`glissando-index` (integer)
: The index of a glissando in its note column.

This grob interface is used in the following graphical object(s): Section 3.1.48 [Glissando], page 430.

3.2.41 grace-spacing-interface

Keep track of durations in a run of grace notes.

User settable properties:

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

Internal properties:

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

This grob interface is used in the following graphical object(s): Section 3.1.49 [GraceSpacing], page 431.

3.2.42 gregorian-ligature-interface

A gregorian ligature.

Internal properties:

`ascendens` (boolean)
: Is this neume of ascending type?

`auctum` (boolean)
: Is this neume liquescentically augmented?

`cavum` (boolean)
: Is this neume outlined?

`context-info` (integer)
: Within a ligature, the final glyph or shape of a head may be affected by the left and/or right neighbour head. `context-info` holds for each head such information about the left and right neighbour, encoded as a bit mask.

`deminutum` (boolean)
: Is this neume deminished?

`descendens` (boolean)
: Is this neume of descendent type?

`inclinatum` (boolean)
: Is this neume an inclinatum?

`linea` (boolean)
: Attach vertical lines to this neume?

`oriscus` (boolean)
: Is this neume an oriscus?

`pes-or-flexa` (boolean)
: Shall this neume be joined with the previous head?

`prefix-set` (number)
: A bit mask that holds all Gregorian head prefixes, such as `\virga` or `\quilisma`.

`quilisma` (boolean)
: Is this neume a quilisma?

`stropha` (boolean)
: Is this neume a stropha?

`virga` (boolean)
: Is this neume a virga?

This grob interface is used in the following graphical object(s): Section 3.1.80 [NoteHead], page 467.

3.2.43 grid-line-interface

A line that is spanned between grid-points.

User settable properties:

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

This grob interface is used in the following graphical object(s): Section 3.1.50 [GridLine], page 431.

3.2.44 grid-point-interface

A spanning point for grid lines.

This grob interface is used in the following graphical object(s): Section 3.1.51 [GridPoint], page 432.

3.2.45 grob-interface

A grob represents a piece of music notation.

All grobs have an X and Y position on the page. These X and Y positions are stored in a relative format, thus they can easily be combined by stacking them, hanging one grob to the side of another, or coupling them into grouping objects.

Each grob has a reference point (a.k.a. parent): The position of a grob is stored relative to that reference point. For example, the X reference point of a staccato dot usually is the note head that it applies to. When the note head is moved, the staccato dot moves along automatically.

A grob is often associated with a symbol, but some grobs do not print any symbols. They take care of grouping objects. For example, there is a separate grob that stacks staves vertically. The Section 3.1.78 [NoteCollision], page 465, object is also an abstract grob: It only moves around chords, but doesn't print anything.

Grobs have properties (Scheme variables) that can be read and set. Two types of them exist: immutable and mutable. Immutable variables define the default style and behavior. They are shared between many objects. They can be changed using `\override` and `\revert`. Mutable properties are variables that are specific to one grob. Typically, lists of other objects, or results from computations are stored in mutable properties. In particular, every call to `ly:grob-set-property!` (or its C++ equivalent) sets a mutable property.

The properties `after-line-breaking` and `before-line-breaking` are dummies that are not user-serviceable.

User settable properties:

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`before-line-breaking` (boolean)
: Dummy property, used to trigger a callback function.

`color` (color)
: The color of this grob.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`footnote-music` (music)
: Music creating a footnote.

`forced-spacing` (number)
: Spacing forced between grobs, used in various ligature engravers.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`id` (string)
: An id string for the grob.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`output-attributes` (list)
: An alist of attributes for the grob, to be included in output files. When the SVG typesetting backend is used, the attributes are assigned to a group (`<g>`) containing all of the stencils that

comprise a given grob. For example, `'((id . 123) (class . foo) (data-whatever . \bar"))` will produce `<g id=\123" class=\foo" data-whatever=\bar"> ... </g>`. In the Postscript backend, where there is no way to group items, the setting of the output-attributes property will have no effect.

`parenthesis-friends` (list)
: A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `'(45 0 0)` rotates by 45 degrees around the center of this object.

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stencil` (stencil)
: The symbol to print.

`transparent` (boolean)
: This makes the grob invisible.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`whiteout` (boolean-or-number)
: If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`. Usually `#f` by default.

`whiteout-style` (symbol)
: Determines the shape of the `whiteout` background. Available are `'outline`, `'rounded-box`, and the default `'box`. There is one exception: Use `'special` for `LyricHyphen`.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

Internal properties:

`axis-group-parent-X` (graphical (layout) object)
: Containing X axis group.

`axis-group-parent-Y` (graphical (layout) object)
: Containing Y axis group.

`cause` (any type)
: Any kind of causation objects (i.e., music, or perhaps translator) that was the cause for this grob.

`cross-staff` (boolean)
: True for grobs whose `Y-extent` depends on inter-staff spacing. The extent is measured relative to the grobs's parent staff (more generally, its `VerticalAxisGroup`) so this boolean flags grobs that are not rigidly fixed to their parent staff. Beams that join notes from two staves are `cross-staff`. Grobs that are positioned around such beams are also `cross-staff`. Grobs that are grouping objects, however, like `VerticalAxisGroups` will not in general be marked `cross-staff` when some of the members of the group are `cross-staff`.

`interfaces` (list)
: A list of symbols indicating the interfaces supported by this object. It is initialized from the `meta` field.

`meta` (list) Provide meta information. It is an alist with the entries `name` and `interfaces`.

`pure-Y-offset-in-progress` (boolean)
: A debugging aid for catching cyclic dependencies.

`staff-symbol` (graphical (layout) object)
: The staff symbol grob that we are in.

This grob interface is used in the following graphical object(s): Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.5 [Ambitus], page 375, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.7 [AmbitusLine], page 377, Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.9 [Arpeggio], page 379, Section 3.1.10 [BalloonTextItem], page 381, Section 3.1.11 [BarLine], page 381, Section 3.1.12 [BarNumber], page 385, Section 3.1.13 [BassFigure], page 386, Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, Section 3.1.23 [BreathingSign], page 394, Section 3.1.24 [ChordName], page 396, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.33 [DotColumn], page 412, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner],

page 420, Section 3.1.41 [Episema], page 422, Section 3.1.42 [Fingering], page 422, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.44 [Flag], page 425, Section 3.1.45 [FootnoteItem], page 426, Section 3.1.46 [FootnoteSpanner], page 427, Section 3.1.47 [FretBoard], page 428, Section 3.1.48 [Glissando], page 430, Section 3.1.49 [GraceSpacing], page 431, Section 3.1.50 [GridLine], page 431, Section 3.1.51 [GridPoint], page 432, Section 3.1.52 [Hairpin], page 432, Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.59 [KievanLigature], page 444, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.63 [LeftEdge], page 447, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.65 [LyricExtender], page 450, Section 3.1.66 [LyricHyphen], page 451, Section 3.1.67 [LyricSpace], page 452, Section 3.1.68 [LyricText], page 453, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.70 [MeasureGrouping], page 456, Section 3.1.71 [MelodyItem], page 457, Section 3.1.72 [MensuralLigature], page 457, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.78 [NoteCollision], page 465, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.81 [NoteName], page 468, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.84 [PaperColumn], page 470, Section 3.1.85 [ParenthesesItem], page 471, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.95 [RestCollision], page 483, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.98 [ScriptRow], page 484, Section 3.1.99 [Slur], page 485, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.102 [SpacingSpanner], page 489, Section 3.1.103 [SpanBar], page 490, Section 3.1.104 [SpanBarStub], page 491, Section 3.1.105 [StaffGrouper], page 491, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.108 [StanzaNumber], page 493, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.116 [System], page 503, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, Section 3.1.121 [TabNoteHead], page 508, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.126 [TimeSignature], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, Section 3.1.133 [UnaCordaPedal], page 525, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, Section 3.1.135 [VaticanaLigature], page 527, Section 3.1.136 [VerticalAlignment], page 528, Section 3.1.137 [VerticalAxisGroup], page 528, Section 3.1.138 [VoiceFollower], page 530, Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.2.46 hairpin-interface

A hairpin crescendo or decrescendo.

User settable properties:

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`broken-bound-padding` (number)
: The amount of padding to insert when a spanner is broken at a line break.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`grow-direction` (direction)
: Crescendo or decrescendo?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

Internal properties:

`adjacent-spanners` (array of grobs)
: An array of directly neighboring dynamic spanners.

`concurrent-hairpins` (array of grobs)
: All concurrent hairpins.

This grob interface is used in the following graphical object(s): Section 3.1.52 [Hairpin], page 432.

3.2.47 hara-kiri-group-spanner-interface

A group spanner that keeps track of interesting items. If it doesn't contain any after line breaking, it removes itself and all its children. Greater control can be exercised via `remove-layer` which can prioritize layers so only the lowest-numbered non-empty layer is retained; make the layer independent of the group; or make it dependent on any other member of the group

User settable properties:

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`remove-layer` (index or symbol)
: When set as a positive integer, the `Keep_alive_together_engraver` removes all `VerticalAxisGroup` grobs with a `remove-layer` larger than the smallest retained `remove-layer`. Set to `#f` to make a layer independent of the `Keep_alive_together_engraver`. Set to `'()`, the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: `#'any` to keep the layer alive with any other layer in the group; `#'above` or `#'below` to keep the layer alive with the context immediately before or after it, respectively.

Internal properties:

`important-column-ranks` (vector)
: A cache of columns that contain `items-worth-living` data.

`items-worth-living` (array of grobs)
: An array of interesting items. If empty in a particular staff, then that staff is erased.

`keep-alive-with` (array of grobs)
: An array of other `VerticalAxisGroup`s. If any of them are alive, then we will stay alive.

`make-dead-when` (array of grobs)
: An array of other `VerticalAxisGroup`s. If any of them are alive, then we will turn dead.

This grob interface is used in the following graphical object(s): Section 3.1.137 [VerticalAxisGroup], page 528.

3.2.48 horizontal-bracket-interface

A horizontal bracket encompassing notes.

User settable properties:

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

Internal properties:

`bracket-text` (graphical (layout) object)
: The text for an analysis bracket.

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

This grob interface is used in the following graphical object(s): Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.83 [OttavaBracket], page 469, and Section 3.1.139 [VoltaBracket], page 531.

3.2.49 horizontal-bracket-text-interface

Label for an analysis bracket.

Internal properties:

`bracket` (graphical (layout) object)
: The bracket for a number.

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

This grob interface is used in the following graphical object(s): Section 3.1.54 [HorizontalBracketText], page 435.

3.2.50 inline-accidental-interface

An inlined accidental (i.e. normal accidentals, cautionary accidentals).

This grob interface is used in the following graphical object(s): Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, and Section 3.1.127 [TrillPitchAccidental], page 518.

3.2.51 instrument-specific-markup-interface

Instrument-specific markup (like fret boards or harp pedal diagrams).

User settable properties:

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a (*property* . *value*) pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.

- `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
- `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
- `paren-padding` – The padding for the parenthesis. Default 0.05.
- `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
- `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
- `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a `(`*`property . value`*`)` pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.
  - `box-height` – Height of each pedal box. Default value 1.0.
  - `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.

- `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
- `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
- `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
- `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`size` (number)
: The ratio of the size of the object to its default size.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.122 [TextScript], page 510.

3.2.52 item-interface

Grobs can be distinguished in their role in the horizontal spacing. Many grobs define constraints on the spacing by their sizes, for example, note heads, clefs, stems, and all other symbols with a fixed shape. These grobs form a subtype called `Item`.

Some items need special treatment for line breaking. For example, a clef is normally only printed at the start of a line (i.e., after a line break). To model this, 'breakable' items (clef, key signature, bar lines, etc.) are copied twice. Then we have three versions of each breakable item: one version if there is no line break, one version that is printed before the line break (at the end of a system), and one version that is printed after the line break.

Whether these versions are visible and take up space is determined by the outcome of the `break-visibility` grob property, which is a function taking a direction (`-1`, `0` or `1`) as an argument. It returns a cons of booleans, signifying whether this grob should be transparent and have no extent.

The following variables for `break-visibility` are predefined:

grob will show:	before break	no break	after break
all-invisible	no	no	no
begin-of-line-visible	no	no	yes
end-of-line-visible	yes	no	no
all-visible	yes	yes	yes
begin-of-line-invisible	yes	yes	no
end-of-line-invisible	no	yes	yes
center-invisible	yes	no	yes

User settable properties:

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

This grob interface is used in the following graphical object(s): Section 3.1.1 [Accidental], page 370, Section 3.1.2 [AccidentalCautionary], page 371, Section 3.1.3 [AccidentalPlacement], page 372, Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.5 [Ambitus], page 375, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.7 [AmbitusLine], page 377, Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.9 [Arpeggio], page 379, Section 3.1.10 [BalloonTextItem], page 381, Section 3.1.11 [BarLine], page 381, Section 3.1.12 [BarNumber], page 385, Section 3.1.13 [BassFigure], page 386, Section 3.1.16 [BassFigureBracket], page 389, Section 3.1.21 [BreakAlignGroup], page 393, Section 3.1.22 [BreakAlignment], page 393, Section 3.1.23 [BreathingSign], page 394, Section 3.1.24 [ChordName], page 396, Section 3.1.25 [Clef], page 397, Section 3.1.26 [ClefModifier], page 400, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.33 [DotColumn], page 412, Section 3.1.34 [Dots], page 412, Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.39 [DynamicText], page 418, Section 3.1.42 [Fingering], page 422, Section 3.1.43 [FingeringColumn], page 424, Section 3.1.44 [Flag], page 425, Section 3.1.45 [FootnoteItem], page 426, Section 3.1.47 [FretBoard], page 428, Section 3.1.50 [GridLine], page 431, Section 3.1.51 [GridPoint], page 432, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.60 [LaissezVibrerTie], page 445, Section 3.1.61 [LaissezVibrerTieColumn], page 446, Section 3.1.63 [LeftEdge], page 447, Section 3.1.68 [LyricText], page 453, Section 3.1.71 [MelodyItem], page 457, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.78 [NoteCollision], page 465, Section 3.1.79 [NoteColumn], page 466, Section 3.1.80 [NoteHead], page 467, Section 3.1.81 [NoteName], page 468, Section 3.1.82 [NoteSpacing], page 468, Section 3.1.84 [PaperColumn], page 470, Section 3.1.85 [ParenthesesItem], page 471, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.92 [RepeatTie], page 480, Section 3.1.93 [RepeatTieColumn], page 481, Section 3.1.94 [Rest], page 481, Section 3.1.95 [RestCollision], page 483, Section 3.1.96 [Script], page 483, Section 3.1.97 [ScriptColumn], page 484, Section 3.1.98 [ScriptRow], page 484, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.103 [SpanBar], page 490, Section 3.1.104 [SpanBarStub], page 491, Section 3.1.106 [StaffSpacing], page 492, Section 3.1.108 [StanzaNumber], page 493, Section 3.1.109 [Stem], page 494, Section 3.1.110 [StemStub], page 496, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.114 [SustainPedal], page 501, Section 3.1.121 [TabNoteHead], page 508, Section 3.1.122 [TextScript], page 510, Section 3.1.126 [TimeSignature], page 515, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.129 [TrillPitchHead], page 520, and Section 3.1.133 [UnaCordaPedal], page 525.

3.2.53 key-cancellation-interface

A key cancellation.

This grob interface is used in the following graphical object(s): Section 3.1.57 [KeyCancellation], page 438.

3.2.54 key-signature-interface

A group of accidentals, to be printed as signature sign.

User settable properties:

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`flat-positions` (list)
: Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`non-default` (boolean)
: Set for manually specified clefs and keys.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping `(`*`name . name`*`)` to distances.

`sharp-positions` (list)
: Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

Internal properties:

`c0-position` (integer)
: An integer indicating the position of middle C.

This grob interface is used in the following graphical object(s): Section 3.1.57 [KeyCancellation], page 438, and Section 3.1.58 [KeySignature], page 441.

3.2.55 kievan-ligature-interface

A kievan ligature.

User settable properties:

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

Internal properties:

`primitive` (integer)
: A pointer to a ligature primitive, i.e., an item similar to a note head that is part of a ligature.

This grob interface is used in the following graphical object(s): Section 3.1.59 [KievanLigature], page 444.

3.2.56 ledger-line-spanner-interface

This spanner draws the ledger lines of a staff. This is a separate grob because it has to process all potential collisions between all note heads. The thickness of ledger lines is controlled by the `ledger-line-thickness` property of the Section 3.1.107 [StaffSymbol], page 493, grob.

User settable properties:

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

Internal properties:

`note-heads` (array of grobs)
: An array of note head grobs.

This grob interface is used in the following graphical object(s): Section 3.1.62 [LedgerLineSpanner], page 446.

3.2.57 ledgered-interface

Objects that need ledger lines, typically note heads. See also Section 3.2.56 [ledger-line-spanner-interface], page 571.

User settable properties:

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

This grob interface is used in the following graphical object(s): Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.80 [NoteHead], page 467, and Section 3.1.129 [TrillPitchHead], page 520.

3.2.58 ligature-bracket-interface

A bracket indicating a ligature in the original edition.

User settable properties:

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline

at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

`width` (dimension, in staff space)
The width of a grob measured in staff space.

This grob interface is not used in any graphical object.

3.2.59 ligature-head-interface

A note head that can become part of a ligature.

This grob interface is used in the following graphical object(s): Section 3.1.80 [NoteHead], page 467.

3.2.60 ligature-interface

A ligature.

This grob interface is not used in any graphical object.

3.2.61 line-interface

Generic line objects. Any object using lines supports this. The property `style` can be `line`, `dashed-line`, `trill`, `dotted-line`, `zigzag` or `none` (a transparent line).

For `dashed-line`, the length of the dashes is tuned with `dash-fraction`. If the latter is set to 0, a dotted line is produced.

User settable properties:

`arrow-length` (number)
Arrow length.

`arrow-width` (number)
Arrow width.

`dash-fraction` (number)
Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number)
The length of one dash together with whitespace. If negative, no line is drawn at all.

`style` (symbol)
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number)
For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

`zigzag-length` (dimension, in staff space)
The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

This grob interface is used in the following graphical object(s): Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.48 [Glissando], page 430, Section 3.1.52 [Hairpin], page 432, Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.123 [TextSpanner], page 512, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.138 [VoiceFollower], page 530, and Section 3.1.139 [VoltaBracket], page 531.

3.2.62 line-spanner-interface

Generic line drawn between two objects, e.g., for use with glissandi.

User settable properties:

`bound-details` (list)
: An alist of properties for determining attachments of spanners to edges.

`extra-dy` (number)
: Slope glissandi this much extra.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`simple-Y` (boolean)
: Should the Y placement of a spanner disregard changes in system heights?

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

Internal properties:

`note-columns` (array of grobs)
: An array of `NoteColumn` grobs.

This grob interface is used in the following graphical object(s): Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.48 [Glissando], page 430, Section 3.1.123 [TextSpanner], page 512, Section 3.1.130 [TrillSpanner], page 521, and Section 3.1.138 [VoiceFollower], page 530.

3.2.63 lyric-extender-interface

The extender is a simple line at the baseline of the lyric that helps show the length of a melisma (a tied or slurred note).

User settable properties:

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`heads` (array of grobs)
: An array of note heads.

This grob interface is used in the following graphical object(s): Section 3.1.65 [LyricExtender], page 450.

3.2.64 lyric-hyphen-interface

A centered hyphen is simply a line between lyrics used to divide syllables.

User settable properties:

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems.

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.66 [LyricHyphen], page 451, and Section 3.1.67 [LyricSpace], page 452.

3.2.65 lyric-interface

Any object that is related to lyrics.

This grob interface is used in the following graphical object(s): Section 3.1.65 [LyricExtender], page 450, and Section 3.1.66 [LyricHyphen], page 451.

3.2.66 lyric-syllable-interface

A single piece of lyrics.

This grob interface is used in the following graphical object(s): Section 3.1.68 [LyricText], page 453.

3.2.67 mark-interface

A rehearsal mark.

This grob interface is used in the following graphical object(s): Section 3.1.90 [RehearsalMark], page 477.

3.2.68 measure-counter-interface

A counter for numbering measures.

User settable properties:

`count-from` (integer)
: The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

Internal properties:

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

This grob interface is used in the following graphical object(s): Section 3.1.69 [MeasureCounter], page 454.

3.2.69 measure-grouping-interface

This object indicates groups of beats. Valid choices for `style` are `bracket` and `triangle`.

User settable properties:

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.70 [Measure-Grouping], page 456.

3.2.70 melody-spanner-interface

Context dependent typesetting decisions.

User settable properties:

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

Internal properties:

`stems` (array of grobs)
: An array of stem objects.

This grob interface is used in the following graphical object(s): Section 3.1.71 [MelodyItem], page 457.

3.2.71 mensural-ligature-interface

A mensural ligature.

User settable properties:

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`add-join` (boolean)
: Is this ligature head-joined with the next one by a vertical line?

`delta-position` (number)
: The vertical position difference.

`flexa-interval` (integer)
: The interval spanned by the two notes of a flexa shape (1 is a second, 7 is an octave).

`head-width` (dimension, in staff space)
: The width of this ligature head.

`ligature-flexa` (boolean)
: request joining note to the previous one in a flexa.

`primitive` (integer)
: A pointer to a ligature primitive, i.e., an item similar to a note head that is part of a ligature.

This grob interface is used in the following graphical object(s): Section 3.1.72 [MensuralLigature], page 457, and Section 3.1.80 [NoteHead], page 467.

3.2.72 metronome-mark-interface

A metronome mark.

This grob interface is used in the following graphical object(s): Section 3.1.73 [MetronomeMark], page 458.

3.2.73 multi-measure-interface

Multi measure rest, and the text or number that is printed over it.

User settable properties:

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

This grob interface is used in the following graphical object(s): Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, and Section 3.1.76 [MultiMeasureRestText], page 463.

3.2.74 multi-measure-rest-interface

A rest that spans a whole number of measures.

User settable properties:

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`hair-thickness` (number)
: Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`max-symbol-separation` (number)
: The maximum distance between symbols making up a church rest.

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`round-up-exceptions` (list)
: A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)
: Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a 3/2 measure.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

`thick-thickness` (number)
: Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`usable-duration-logs` (list)
: List of `duration-log`s that can be used in typesetting the grob.

Internal properties:

`space-increment` (dimension, in staff space)
: The amount by which the total duration of a multimeasure rest affects horizontal spacing. Each doubling of the duration adds `space-increment` to the length of the bar.

This grob interface is used in the following graphical object(s): Section 3.1.74 [MultiMeasureRest], page 459, and Section 3.1.86 [PercentRepeat], page 472.

3.2.75 note-collision-interface

An object that handles collisions between notes with different stem directions and horizontal shifts. Most of the interesting properties are to be set in Section 3.2.76 [note-column-interface], page 579: these are `force-hshift` and `horizontal-shift`.

User settable properties:

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Section "note-collision-interface" in *Internals Reference*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`note-collision-threshold` (dimension, in staff space)
: Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

Internal properties:

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.78 [NoteCollision], page 465.

3.2.76 note-column-interface

Stem and noteheads combined.

User settable properties:

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Section "note-collision-interface" in *Internals Reference*.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Section "note-collision-interface" in *Internals Reference*.

`ignore-collision` (boolean)
: If set, don't do note collision resolution on this `NoteColumn`.

Internal properties:

`note-heads` (array of grobs)
: An array of note head grobs.

`rest` (graphical (layout) object)
: A pointer to a `Rest` object.

`rest-collision` (graphical (layout) object)
: A rest collision that a rest is in.

`stem` (graphical (layout) object)
: A pointer to a `Stem` object.

This grob interface is used in the following graphical object(s): Section 3.1.79 [NoteColumn], page 466.

3.2.77 note-head-interface

A note head. There are many possible values for `style`. For a complete list, see Section "Note head styles" in *Notation Reference*.

User settable properties:

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`ignore-ambitus` (boolean)
: If set, don't consider this notehead for ambitus calculation.

`ledger-positions` (list)
: Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

Internal properties:

`accidental-grob` (graphical (layout) object)
: The accidental for this note.

This grob interface is used in the following graphical object(s): Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.80 [NoteHead], page 467, Section 3.1.121 [TabNoteHead], page 508, and Section 3.1.128 [TrillPitchGroup], page 519.

3.2.78 note-name-interface

Note names.

This grob interface is used in the following graphical object(s): Section 3.1.81 [NoteName], page 468.

3.2.79 note-spacing-interface

This object calculates spacing wishes for individual voices.

User settable properties:

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

Internal properties:

`left-items` (array of grobs)
: Grobs organized on the left by a spacing object.

`right-items` (array of grobs)
: Grobs organized on the right by a spacing object.

This grob interface is used in the following graphical object(s): Section 3.1.82 [NoteSpacing], page 468.

3.2.80 number-interface

Numbers.

User settable properties:

`number-type` (symbol)
: Numbering style. Choices include `roman-lower`, `roman-upper` and `arabic`.

This grob interface is used in the following graphical object(s): Section 3.1.112 [StringNumber], page 498.

3.2.81 only-prebreak-interface

Kill this grob after the line breaking process.

This grob interface is not used in any graphical object.

3.2.82 ottava-bracket-interface

An ottava bracket.

User settable properties:

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

This grob interface is used in the following graphical object(s): Section 3.1.83 [OttavaBracket], page 469.

3.2.83 outside-staff-axis-group-interface

A vertical axis group on which outside-staff skyline calculations are done.

User settable properties:

`outside-staff-placement-directive` (symbol)
: One of four directives telling how outside staff objects should be placed.

- `left-to-right-greedy` – Place each successive grob from left to right.
- `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
- `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
- `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

Internal properties:

`vertical-skyline-elements` (array of grobs)
: An array of grobs used to create vertical skylines.

This grob interface is used in the following graphical object(s): Section 3.1.18 [BassFigureLine], page 389, Section 3.1.116 [System], page 503, and Section 3.1.137 [VerticalAxisGroup], page 528.

3.2.84 outside-staff-interface

A grob that could be placed outside staff.

User settable properties:

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

This grob interface is used in the following graphical object(s): Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.12 [BarNumber], page 385, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.23 [BreathingSign], page 394, Section 3.1.24 [ChordName], page 396, Section 3.1.26 [ClefModifier], page 400, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.39 [DynamicText], page 418, Section 3.1.42 [Fingering], page 422, Section 3.1.47 [FretBoard], page 428, Section 3.1.52 [Hairpin], page 432, Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.70 [MeasureGrouping], page 456, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.96 [Script], page 483, Section 3.1.99 [Slur], page 485, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.2.85 paper-column-interface

`Paper_column` objects form the top-most X parents for items. There are two types of columns: musical and non-musical, to which musical and non-musical objects are attached respectively. The spacing engine determines the X positions of these objects.

They are numbered, the first (leftmost) is column 0. Numbering happens before line breaking, and columns are not renumbered after line breaking. Since many columns go unused, you should only use the rank field to get ordering information. Two adjacent columns may have non-adjacent numbers.

User settable properties:

`between-cols` (pair)
: Where to attach a loose column to.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`labels` (list)
: List of labels (symbols) placed on a column.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`when` (moment)
: Global time step associated with this column.

Internal properties:

`bounded-by-me` (array of grobs)
: An array of spanners that have this column as start/begin point. Only columns that have grobs or act as bounds are spaced.

`grace-spacing` (graphical (layout) object)
: A run of grace notes.

`maybe-loose` (boolean)
: Used to mark a breakable column that is loose if and only if it is in the middle of a line.

`spacing` (graphical (layout) object)
: The spacing spanner governing this section.

This grob interface is used in the following graphical object(s): Section 3.1.77 [NonMusicalPaperColumn], page 464, and Section 3.1.84 [PaperColumn], page 470.

3.2.86 parentheses-interface

Parentheses for other objects.

User settable properties:

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`stencils` (list)
: Multiple stencils, used as intermediate value.

This grob interface is used in the following graphical object(s): Section 3.1.85 [ParenthesesItem], page 471, and Section 3.1.128 [TrillPitchGroup], page 519.

3.2.87 percent-repeat-interface

Beat, Double and single measure repeats.

User settable properties:

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, and Section 3.1.91 [RepeatSlash], page 479.

3.2.88 percent-repeat-item-interface

Repeats that look like percent signs.

User settable properties:

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.35 [DoublePercentRepeat], page 413, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.37 [DoubleRepeatSlash], page 416, and Section 3.1.91 [RepeatSlash], page 479.

3.2.89 piano-pedal-bracket-interface

The bracket of the piano pedal. It can be tuned through the regular bracket properties.

User settable properties:

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*left-height* `.` *right-height*`)`.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

Internal properties:

`pedal-text` (graphical (layout) object)
: A pointer to the text of a mixed-style piano pedal.

This grob interface is used in the following graphical object(s): Section 3.1.89 [PianoPedalBracket], page 476.

3.2.90 piano-pedal-interface

A piano pedal sign.

This grob interface is used in the following graphical object(s): Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.114 [SustainPedal], page 501, Section 3.1.115 [SustainPedalLineSpanner], page 502, and Section 3.1.134 [UnaCordaPedalLineSpanner], page 526.

3.2.91 piano-pedal-script-interface

A piano pedal sign, fixed size.

This grob interface is used in the following graphical object(s): Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.114 [SustainPedal], page 501, and Section 3.1.133 [UnaCordaPedal], page 525.

3.2.92 pitched-trill-interface

A note head to indicate trill pitches.

Internal properties:

`accidental-grob` (graphical (layout) object)
: The accidental for this note.

This grob interface is used in the following graphical object(s): Section 3.1.129 [TrillPitchHead], page 520.

3.2.93 pure-from-neighbor-interface

A collection of routines to allow for objects' pure heights and heights to be calculated based on the heights of the objects' neighbors.

Internal properties:

`neighbors` (array of grobs)
: The X-axis neighbors of a grob. Used by the pure-from-neighbor-interface to determine various grob heights.

`pure-relevant-grobs` (array of grobs)
: All the grobs (items and spanners) that are relevant for finding the `pure-Y-extent`

`pure-Y-common` (graphical (layout) object)
: A cache of the `common_refpoint_of_array` of the `elements` grob set.

This grob interface is used in the following graphical object(s): Section 3.1.11 [BarLine], page 381, Section 3.1.25 [Clef], page 397, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.104 [SpanBarStub], page 491, and Section 3.1.126 [TimeSignature], page 515.

3.2.94 rest-collision-interface

Move ordinary rests (not multi-measure nor pitched rests) to avoid conflicts.

User settable properties:

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

Internal properties:

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

This grob interface is used in the following graphical object(s): Section 3.1.95 [RestCollision], page 483.

3.2.95 rest-interface

A rest symbol. The property `style` can be `default`, `mensural`, `neomensural` or `classical`.

User settable properties:

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`voiced-position` (number)
: The staff-position of a voiced `Rest`, negative if the rest has `direction` `DOWN`.

This grob interface is used in the following graphical object(s): Section 3.1.74 [MultiMeasureRest], page 459, and Section 3.1.94 [Rest], page 481.

3.2.96 rhythmic-grob-interface

Any object with a duration. Used to determine which grobs are interesting enough to maintain a hara-kiri staff.

This grob interface is used in the following graphical object(s): Section 3.1.13 [BassFigure], page 386, Section 3.1.24 [ChordName], page 396, Section 3.1.28 [ClusterSpannerBeacon], page 402, Section 3.1.37 [DoubleRepeatSlash], page 416, Section 3.1.47 [FretBoard], page 428, Section 3.1.68 [LyricText], page 453, Section 3.1.80 [NoteHead], page 467, Section 3.1.91 [RepeatSlash], page 479, Section 3.1.94 [Rest], page 481, and Section 3.1.121 [TabNoteHead], page 508.

3.2.97 rhythmic-head-interface

Note head or rest.

User settable properties:

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

Internal properties:

`dot` (graphical (layout) object)
: A reference to a `Dots` object.

`stem` (graphical (layout) object)
: A pointer to a `Stem` object.

This grob interface is used in the following graphical object(s): Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.80 [NoteHead], page 467, Section 3.1.94 [Rest], page 481, Section 3.1.121 [TabNoteHead], page 508, and Section 3.1.129 [TrillPitchHead], page 520.

3.2.98 script-column-interface

An interface that sorts scripts according to their `script-priority` and `outside-staff-priority`.

Internal properties:

`scripts` (array of grobs)
: An array of `Script` objects.

This grob interface is used in the following graphical object(s): Section 3.1.97 [ScriptColumn], page 484, and Section 3.1.98 [ScriptRow], page 484.

3.2.99 script-interface

An object that is put above or below a note.

User settable properties:

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`toward-stem-shift-in-column` (number)
: Amount by which a script is shifted toward the stem if its direction coincides with the stem direction and it is associated with a `ScriptColumn` object. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

Internal properties:

`direction-source` (graphical (layout) object)
: In case `side-relative-direction` is set, which grob to get the direction from.

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

`script-column` (graphical (layout) object)
: A `ScriptColumn` associated with a `Script` object.

`script-stencil` (pair)
: A pair `(`*`type`* `.` *`arg`*`)` which acts as an index for looking up a `Stencil` object.

`slur` (graphical (layout) object)
: A pointer to a `Slur` object.

This grob interface is used in the following graphical object(s): Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.39 [DynamicText], page 418, and Section 3.1.96 [Script], page 483.

3.2.100 self-alignment-interface

Position this object on itself and/or on its parent. To this end, the following functions are provided:

`Self_alignment_interface::[xy]_aligned_on_self`
: Align self on reference point, using `self-alignment-X` and `self-alignment-Y`.

`Self_alignment_interface::aligned_on_[xy]_parent`
`Self_alignment_interface::centered_on_[xy]_parent`
: Shift the object so its own reference point is centered on the extent of the parent

User settable properties:

`parent-alignment-X` (number)
: Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`parent-alignment-Y` (number)
: Like `parent-alignment-X` but for the Y axis.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`X-align-on-main-noteheads` (boolean)
: If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

This grob interface is used in the following graphical object(s): Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.12 [BarNumber], page 385, Section 3.1.26 [ClefModifier], page 400, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.39 [DynamicText], page 418, Section 3.1.42 [Fingering], page 422, Section 3.1.50 [GridLine], page 431, Section 3.1.52 [Hairpin], page 432, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436,

Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.68 [LyricText], page 453, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.96 [Script], page 483, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.111 [StemTremolo], page 497, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.114 [SustainPedal], page 501, Section 3.1.122 [TextScript], page 510, and Section 3.1.133 [UnaCordaPedal], page 525.

3.2.101 semi-tie-column-interface

The interface for a column of l.v. (laissez vibrer) ties.

User settable properties:

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`tie-configuration` (list)
: List of (*position . dir*) pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

Internal properties:

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

`ties` (array of grobs)
: A grob array of `Tie` objects.

This grob interface is used in the following graphical object(s): Section 3.1.61 [LaissezVibrerTieColumn], page 446, and Section 3.1.93 [RepeatTieColumn], page 481.

3.2.102 semi-tie-interface

A tie which is only connected to a note head on one side. The following properties may be set in the `details` list:

`height-limit`
: Maximum tie height: The longer the tie, the closer it is to this height.

`ratio`
: Parameter for tie shape. The higher this number, the quicker the tie attains its `height-limit`.

User settable properties:

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`note-head` (graphical (layout) object)
: A single note head.

This grob interface is used in the following graphical object(s): Section 3.1.60 [LaissezVibrerTie], page 445, and Section 3.1.92 [RepeatTie], page 480.

3.2.103 separation-item-interface

Item that computes widths to generate spacing rods.

User settable properties:

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

Internal properties:

`conditional-elements` (array of grobs)
: Internal use only.

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

This grob interface is used in the following graphical object(s): Section 3.1.77 [NonMusicalPaperColumn], page 464, Section 3.1.79 [NoteColumn], page 466, and Section 3.1.84 [PaperColumn], page 470.

3.2.104 side-position-interface

Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

The routine also takes the size of the staff into account if `staff-padding` is set. If undefined, the staff symbol is ignored.

User settable properties:

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`horizon-padding` (number)
: The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`side-axis` (number)
: If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`slur-padding` (number)
: Extra distance between slur and script.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`use-skylines` (boolean)
: Should skylines be used for side positioning?

Internal properties:

`quantize-position` (boolean)
: If set, a vertical alignment is aligned to be within staff spaces.

`side-support-elements` (array of grobs)
: The side support, an array of grobs.

This grob interface is used in the following graphical object(s): Section 3.1.4 [AccidentalSuggestion], page 373, Section 3.1.6 [AmbitusAccidental], page 376, Section 3.1.9 [Arpeggio], page 379, Section 3.1.12 [BarNumber], page 385, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.26 [ClefModifier], page 400, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.41 [Episema], page 422, Section 3.1.42 [Fingering], page 422, Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.70 [MeasureGrouping], page 456, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.96 [Script], page 483, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.108 [StanzaNumber], page 493, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, Section 3.1.122 [TextScript], page 510, Section 3.1.123 [TextSpanner], page 512, Section 3.1.127 [TrillPitchAccidental], page 518, Section 3.1.128 [TrillPitchGroup], page 519, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.2.105 slur-interface

A slur. The following properties may be set in the `details` list.

`region-size`
: Size of region (in staff spaces) for determining potential endpoints in the Y direction.

`head-encompass-penalty`
: Demerit to apply when note heads collide with a slur.

`stem-encompass-penalty`
: Demerit to apply when stems collide with a slur.

`edge-attraction-factor`
: Factor used to calculate the demerit for distances between slur endpoints and their corresponding base attachments.

`same-slope-penalty`
: Demerit for slurs with attachment points that are horizontally aligned.

`steeper-slope-factor`
: Factor used to calculate demerit only if this slur is not broken.

`non-horizontal-penalty`
: Demerit for slurs with attachment points that are not horizontally aligned.

`max-slope`
: The maximum slope allowed for this slur.

`max-slope-factor`
: Factor that calculates demerit based on the max slope.

`free-head-distance`
: The amount of vertical free space that must exist between a slur and note heads.

`absolute-closeness-measure`
: Factor to calculate demerit for variance between a note head and slur.

`extra-object-collision-penalty`
: Factor to calculate demerit for extra objects that the slur encompasses, including accidentals, fingerings, and tuplet numbers.

`accidental-collision`
: Factor to calculate demerit for `Accidental` objects that the slur encompasses. This property value replaces the value of `extra-object-collision-penalty`.

`extra-encompass-free-distance`
: The amount of vertical free space that must exist between a slur and various objects it encompasses, including accidentals, fingerings, and tuplet numbers.

`extra-encompass-collision-distance`
: This detail is currently unused.

`head-slur-distance-factor`
: Factor to calculate demerit for variance between a note head and slur.

`head-slur-distance-max-ratio`
: The maximum value for the ratio of distance between a note head and slur.

`gap-to-staffline-inside`
: Minimum gap inside the curve of the slur where the slur is parallel to a staffline.

`gap-to-staffline-outside`
: Minimum gap outside the curve of the slur where the slur is parallel to a staffline.

`free-slur-distance`
: The amount of vertical free space that must exist between adjacent slurs. This subproperty only works for `PhrasingSlur`.

`edge-slope-exponent`
: Factor used to calculate the demerit for the slope of a slur near its endpoints; a larger value yields a larger demerit.

User settable properties:

`annotation` (string)
: Annotate a grob for debug purposes.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`inspect-index` (integer)
: If debugging is set, set beam and slur configuration to this index, and print the respective scores.

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`positions` (pair of numbers)
: Pair of staff coordinates (*`left . right`*), where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

Internal properties:

`encompass-objects` (array of grobs)
: Objects that a slur should avoid in addition to notes and stems.

`note-columns` (array of grobs)
: An array of `NoteColumn` grobs.

This grob interface is used in the following graphical object(s): Section 3.1.88 [PhrasingSlur], page 474, and Section 3.1.99 [Slur], page 485.

3.2.106 spaceable-grob-interface

A layout object that takes part in the spacing problem.

User settable properties:

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

Internal properties:

`ideal-distances` (list)
: `(obj . (dist . strength))` pairs.

`left-neighbor` (graphical (layout) object)
: The right-most column that has a spacing-wish for this column.

`minimum-distances` (list)
: A list of rods that have the format `(obj . dist)`.

`right-neighbor` (graphical (layout) object)
: See `left-neighbor`.

`spacing-wishes` (array of grobs)
: An array of note spacing or staff spacing objects.

This grob interface is used in the following graphical object(s): Section 3.1.77 [NonMusicalPaperColumn], page 464, and Section 3.1.84 [PaperColumn], page 470.

3.2.107 spacing-interface

This object calculates the desired and minimum distances between two columns.

Internal properties:

`left-items` (array of grobs)
: Grobs organized on the left by a spacing object.

`right-items` (array of grobs)
: Grobs organized on the right by a spacing object.

This grob interface is used in the following graphical object(s): Section 3.1.82 [NoteSpacing], page 468, and Section 3.1.106 [StaffSpacing], page 492.

3.2.108 spacing-options-interface

Supports setting of spacing variables.

User settable properties:

`shortest-duration-space` (number)
: Start with this multiple of `spacing-increment` space for the shortest duration. See also Section “spacing-spanner-interface” in *Internals Reference*.

`spacing-increment` (dimension, in staff space)
: The unit of length for note-spacing. Typically, the width of a note head. See also Section "spacing-spanner-interface" in *Internals Reference*.

This grob interface is used in the following graphical object(s): Section 3.1.49 [GraceSpacing], page 431, and Section 3.1.102 [SpacingSpanner], page 489.

3.2.109 spacing-spanner-interface

The space taken by a note is dependent on its duration. Doubling a duration adds `spacing-increment` to the space. The most common shortest note gets `shortest-duration-space`. Notes that are even shorter are spaced proportonial to their duration.

Typically, the increment is the width of a black note head. In a piece with lots of 8th notes, and some 16th notes, the eighth note gets a 2 note heads width (i.e., the space following a note is a 1 note head width). A 16th note is followed by 0.5 note head width. The quarter note is followed by 3 NHW, the half by 4 NHW, etc.

User settable properties:

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`base-shortest-duration` (moment)
: Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`shortest-duration-space` (number)
: Start with this multiple of `spacing-increment` space for the shortest duration. See also Section "spacing-spanner-interface" in *Internals Reference*.

`spacing-increment` (dimension, in staff space)
: The unit of length for note-spacing. Typically, the width of a note head. See also Section "spacing-spanner-interface" in *Internals Reference*.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

This grob interface is used in the following graphical object(s): Section 3.1.102 [SpacingSpanner], page 489.

3.2.110 span-bar-interface

A bar line that is spanned between other barlines. This interface is used for bar lines that connect different staves.

User settable properties:

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

Internal properties:

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

`pure-relevant-grobs` (array of grobs)
: All the grobs (items and spanners) that are relevant for finding the `pure-Y-extent`

`pure-relevant-items` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-relevant-spanners` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-Y-common` (graphical (layout) object)
: A cache of the `common_refpoint_of_array` of the `elements` grob set.

This grob interface is used in the following graphical object(s): Section 3.1.103 [SpanBar], page 490.

3.2.111 spanner-interface

Some objects are horizontally spanned between objects. For example, slurs, beams, ties, etc. These grobs form a subtype called `Spanner`. All spanners have two span points (these must be `Item` objects), one on the left and one on the right. The left bound is also the X reference point of the spanner.

User settable properties:

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-after-break` (dimension, in staff space)
: If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance to the notehead.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`spanner-id` (index or symbol)
: An identifier to distinguish concurrent spanners.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

Internal properties:

`spanner-broken` (boolean)
: Indicates whether spanner alignment should be broken after the current spanner.

This grob interface is used in the following graphical object(s): Section 3.1.14 [BassFigureAlignment], page 387, Section 3.1.15 [BassFigureAlignmentPositioning], page 387, Section 3.1.17 [BassFigureContinuation], page 389, Section 3.1.18 [BassFigureLine], page 389, Section 3.1.19 [Beam], page 390, Section 3.1.20 [BendAfter], page 392, Section 3.1.27 [ClusterSpanner], page 402, Section 3.1.38 [DynamicLineSpanner], page 417, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.41 [Episema], page 422, Section 3.1.46 [FootnoteSpanner], page 427, Section 3.1.48 [Glissando], page 430, Section 3.1.49 [GraceSpacing], page 431, Section 3.1.52 [Hairpin], page 432, Section 3.1.53 [HorizontalBracket], page 434, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436, Section 3.1.59 [KievanLigature], page 444, Section 3.1.62 [LedgerLineSpanner], page 446, Section 3.1.64 [LigatureBracket], page 449, Section 3.1.65 [LyricExtender], page 450, Section 3.1.66 [LyricHyphen], page 451, Section 3.1.67 [LyricSpace], page 452, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.70 [MeasureGrouping], page 456, Section 3.1.72 [MensuralLigature], page 457, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.86 [PercentRepeat], page 472, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.88 [PhrasingSlur], page 474, Section 3.1.89 [PianoPedalBracket], page 476, Section 3.1.99 [Slur], page 485, Section 3.1.101 [SostenutoPedalLineSpanner], page 488, Section 3.1.102 [SpacingSpanner], page 489, Section 3.1.105 [StaffGrouper], page 491, Section 3.1.107 [StaffSymbol], page 493, Section 3.1.115 [SustainPedalLineSpanner], page 502, Section 3.1.116 [System], page 503, Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, Section 3.1.120 [SystemStartSquare], page 507, Section 3.1.123 [TextSpanner], page 512, Section 3.1.124 [Tie], page 513, Section 3.1.125 [TieColumn], page 515, Section 3.1.130 [TrillSpanner], page 521, Section 3.1.131 [TupletBracket], page 522, Section 3.1.132 [TupletNumber], page 524, Section 3.1.134 [UnaCordaPedalLineSpanner], page 526, Section 3.1.135 [VaticanaLigature], page 527, Section 3.1.136 [VerticalAlignment], page 528, Section 3.1.137 [VerticalAxisGroup], page 528, Section 3.1.138 [VoiceFollower], page 530, Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.2.112 staff-grouper-interface

A grob that collects staves together.

User settable properties:

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.

- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

This grob interface is used in the following graphical object(s): Section 3.1.105 [StaffGrouper], page 491.

3.2.113 staff-spacing-interface

This object calculates spacing details from a breakable symbol (left) to another object. For example, it takes care of optical spacing from a bar line to a note.

User settable properties:

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

This grob interface is used in the following graphical object(s): Section 3.1.106 [StaffSpacing], page 492.

3.2.114 staff-symbol-interface

This spanner draws the lines of a staff. A staff symbol defines a vertical unit, the *staff space*. Quantities that go by a half staff space are called *positions*. The center (i.e., middle line or space) is position 0. The length of the symbol may be set by hand through the `width` property.

User settable properties:

`break-align-symbols` (list)
: A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`ledger-extra` (dimension, in staff space)
: Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)
: Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`ledger-positions-function` (any type)
: A quoted Scheme procedure that takes a `StaffSymbol` grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

This grob interface is used in the following graphical object(s): Section 3.1.107 [StaffSymbol], page 493.

3.2.115 staff-symbol-referencer-interface

An object whose Y position is meant relative to a staff symbol. These usually have `Staff_symbol_referencer::callback` in their `Y-offset-callbacks`.

User settable properties:

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

This grob interface is used in the following graphical object(s): Section 3.1.8 [AmbitusNoteHead], page 378, Section 3.1.9 [Arpeggio], page 379, Section 3.1.19 [Beam], page 390, Section 3.1.25 [Clef], page 397, Section 3.1.30 [CueClef], page 404, Section 3.1.31 [CueEndClef], page 407, Section 3.1.32 [Custos], page 410, Section 3.1.34 [Dots], page 412, Section 3.1.57 [KeyCancellation], page 438, Section 3.1.58 [KeySignature], page 441, Section 3.1.74 [MultiMeasureRest], page 459, Section 3.1.80 [NoteHead], page 467, Section 3.1.94 [Rest], page 481, Section 3.1.121 [TabNoteHead], page 508, and Section 3.1.129 [TrillPitchHead], page 520.

3.2.116 stanza-number-interface

A stanza number, to be put in from of a lyrics line.

This grob interface is used in the following graphical object(s): Section 3.1.108 [StanzaNumber], page 493.

3.2.117 stem-interface

The stem represents the graphical stem. In addition, it internally connects note heads, beams, and tremolos. Rests and whole notes have invisible stems.

The following properties may be set in the `details` list.

`beamed-lengths`
: List of stem lengths given beam multiplicity.

`beamed-minimum-free-lengths`
: List of normal minimum free stem lengths (chord to beams) given beam multiplicity.

`beamed-extreme-minimum-free-lengths`
: List of extreme minimum free stem lengths (chord to beams) given beam multiplicity.

`lengths` Default stem lengths. The list gives a length for each flag count.

`stem-shorten`
: How much a stem in a forced direction should be shortened. The list gives an amount depending on the number of flags and beams.

User settable properties:

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
: A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
: The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`default-direction` (direction)
: Direction determined by note head positions.

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP=1`, `DOWN=-1`, `LEFT=-1`, `RIGHT=1`, `CENTER=0`.

`double-stem-separation` (number)
: The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`note-collision-threshold` (dimension, in staff space)
: Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`stem-begin-position` (number)
: User override for the begin position of a stem.

`stemlet-length` (number)
: How long should be a stem over a rest?

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

Internal properties:

`beam` (graphical (layout) object)
: A pointer to the beam, if applicable.

`flag` (graphical (layout) object)
: A pointer to a `Flag` object.

`melody-spanner` (graphical (layout) object)
: The `MelodyItem` object for a stem.

`note-heads` (array of grobs)
: An array of note head grobs.

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

`rests` (array of grobs)
: An array of rest objects.

`stem-info` (pair)
: A cache of stem parameters.

`tremolo-flag` (graphical (layout) object)
: The tremolo object on a stem.

`tuplet-start` (boolean)
: Is stem at the start of a tuplet?

This grob interface is used in the following graphical object(s): Section 3.1.109 [Stem], page 494.

3.2.118 stem-tremolo-interface

A beam slashing a stem to indicate a tremolo. The property `shape` can be `beam-like` or `rectangle`.

User settable properties:

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
: Width of the tremolo sign.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`flag-count` (number)
: The number of tremolo beams.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`shape` (symbol)
: This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

`slope` (number)
: The slope of this object.

Internal properties:

`stem` (graphical (layout) object)
: A pointer to a `Stem` object.

This grob interface is used in the following graphical object(s): Section 3.1.111 [StemTremolo], page 497.

3.2.119 string-number-interface

A string number instruction.

This grob interface is used in the following graphical object(s): Section 3.1.112 [StringNumber], page 498.

3.2.120 stroke-finger-interface

A right hand finger instruction.

User settable properties:

`digit-names` (vector)
: Names for string finger digits.

This grob interface is used in the following graphical object(s): Section 3.1.113 [StrokeFinger], page 500.

3.2.121 system-interface

This is the top-level object: Each object in a score ultimately has a `System` object as its X and Y parent.

User settable properties:

`labels` (list)
: List of labels (symbols) placed on a column.

Internal properties:

`all-elements` (array of grobs)
: An array of all grobs in this line. Its function is to protect objects from being garbage collected.

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

`footnote-stencil` (stencil)
: The stencil of a system's footnotes.

`footnotes-after-line-breaking` (array of grobs)
: Footnote grobs of a broken system.

`footnotes-before-line-breaking` (array of grobs)
: Footnote grobs of a whole system.

`in-note-direction` (direction)
: Direction to place in-notes above a system.

`in-note-padding` (number)
: Padding between in-notes.

`in-note-stencil` (stencil)
: The stencil of a system's in-notes.

`pure-Y-extent` (pair of numbers)
: The estimated height of a system.

`vertical-alignment` (graphical (layout) object)
: The VerticalAlignment in a System.

This grob interface is used in the following graphical object(s): Section 3.1.116 [System], page 503.

3.2.122 system-start-delimiter-interface

The brace, bracket or bar in front of the system. The following values for `style` are recognized:

`bracket` A thick bracket, normally used to group similar instruments in a score. Default for `StaffGroup`. `SystemStartBracket` uses this style.

`brace` A 'piano style' brace normally used for an instrument that uses two staves. The default style for `GrandStaff`. `SystemStartBrace` uses this style.

`bar-line` A simple line between the staves in a score. Default for staves enclosed in `<<` and `>>`. `SystemStartBar` uses this style.

`line-bracket`
: A simple square, normally used for subgrouping instruments in a score. `SystemStartSquare` uses this style.

See also `input/regression/system-start-nesting.ly`.

User settable properties:

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): Section 3.1.117 [SystemStartBar], page 504, Section 3.1.118 [SystemStartBrace], page 505, Section 3.1.119 [SystemStartBracket], page 506, and Section 3.1.120 [SystemStartSquare], page 507.

3.2.123 system-start-text-interface

Text in front of the system.

User settable properties:

`long-text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

This grob interface is used in the following graphical object(s): Section 3.1.55 [InstrumentName], page 436.

3.2.124 tab-note-head-interface

A note head in tablature.

User settable properties:

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

Internal properties:

`display-cautionary` (boolean)
: Should the grob be displayed as a cautionary grob?

`span-start` (boolean)
: Is the note head at the start of a spanner?

This grob interface is used in the following graphical object(s): Section 3.1.121 [TabNoteHead], page 508.

3.2.125 text-interface

A Scheme markup text, see Section "Formatting text" in *Notation Reference* and Section "New markup command definition" in *Extending*.

There are two important commands: `ly:text-interface::print`, which is a grob callback, and `ly:text-interface::interpret-markup`.

User settable properties:

`baseline-skip` (dimension, in staff space)
: Distance between base lines of multiple lines of text.

`flag-style` (symbol)
: The style of the flag to be used with `MetronomeMark`. Available are `'modern-straight-flag`, `'old-straight-flag`, `flat-flag`, `mensural` and `'default`

`replacement-alist` (list)
: Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

`text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

This grob interface is used in the following graphical object(s): Section 3.1.10 [BalloonTextItem], page 381, Section 3.1.12 [BarNumber], page 385, Section 3.1.13 [BassFigure], page 386, Section 3.1.23 [BreathingSign], page 394, Section 3.1.24 [ChordName], page 396, Section 3.1.26 [ClefModifier], page 400, Section 3.1.29 [CombineTextScript], page 402, Section 3.1.36 [DoublePercentRepeatCounter], page 414, Section 3.1.39 [DynamicText], page 418, Section 3.1.40 [DynamicTextSpanner], page 420, Section 3.1.42 [Fingering], page 422, Section 3.1.45 [FootnoteItem], page 426, Section 3.1.46 [FootnoteSpanner], page 427, Section 3.1.54 [HorizontalBracketText], page 435, Section 3.1.55 [InstrumentName], page 436, Section 3.1.56 [InstrumentSwitch], page 437, Section 3.1.68 [LyricText], page 453, Section 3.1.69 [MeasureCounter], page 454, Section 3.1.73 [MetronomeMark], page 458, Section 3.1.75 [MultiMeasureRestNumber], page 461, Section 3.1.76 [MultiMeasureRestText], page 463, Section 3.1.81 [NoteName], page 468, Section 3.1.83 [OttavaBracket], page 469, Section 3.1.87 [PercentRepeatCounter], page 473, Section 3.1.90 [RehearsalMark], page 477, Section 3.1.100 [SostenutoPedal], page 487, Section 3.1.108 [StanzaNumber], page 493, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, Section 3.1.114 [SustainPedal], page 501, Section 3.1.121 [TabNoteHead], page 508, Section 3.1.122 [TextScript], page 510, Section 3.1.132 [TupletNumber], page 524, Section 3.1.133 [UnaCordaPedal], page 525, and Section 3.1.139 [VoltaBracket], page 531.

3.2.126 text-script-interface

An object that is put above or below a note.

User settable properties:

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

Internal properties:

`slur` (graphical (layout) object)
: A pointer to a `Slur` object.

This grob interface is used in the following graphical object(s): Section 3.1.29 [CombineTextScript], page 402, Section 3.1.42 [Fingering], page 422, Section 3.1.112 [StringNumber], page 498, Section 3.1.113 [StrokeFinger], page 500, and Section 3.1.122 [TextScript], page 510.

3.2.127 tie-column-interface

Object that sets directions of multiple ties in a tied chord.

User settable properties:

`tie-configuration` (list)
: List of (*position* . *dir*) pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

Internal properties:

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

`ties` (array of grobs)
: A grob array of `Tie` objects.

This grob interface is used in the following graphical object(s): Section 3.1.125 [TieColumn], page 515.

3.2.128 tie-interface

A tie - a horizontal curve connecting two noteheads.

The following properties may be set in the `details` list.

`height-limit`
: The maximum height allowed for this tie.

`ratio`
: Parameter for tie shape. The higher this number, the quicker the slur attains its height-limit.

`between-length-limit`
: This detail is currently unused.

`wrong-direction-offset-penalty`
: Demerit for ties that are offset in the wrong direction.

`min-length`
: If the tie is shorter than this amount (in staff-spaces) an increasingly large length penalty is incurred.

`min-length-penalty-factor`
: Demerit factor for tie lengths shorter than `min-length`.

`center-staff-line-clearance`
: If the center of the tie is closer to a staff line than this amount, an increasingly large staff line collision penalty is incurred.

`tip-staff-line-clearance`
: If the tips of the tie are closer to a staff line than this amount, an increasingly large staff line collision penalty is incurred.

`staff-line-collision-penalty`
: Demerit factor for ties whose tips or center come close to staff lines.

`dot-collision-clearance`
: If the tie comes closer to a dot than this amount, an increasingly large dot collision penalty is incurred.

`dot-collision-penalty`
: Demerit factor for ties which come close to dots.

`note-head-gap`
: The distance (in staff-spaces) by which the ends of the tie are offset horizontally from the center line through the note head.

`stem-gap`
: The distance (in staff-spaces) by which the ends of the tie are offset horizontally from a stem which is on the same side of the note head as the tie.

`tie-column-monotonicity-penalty`
: Demerit if the y-position of this tie in the set of ties being considered is less than the y-position of the previous tie.

`tie-tie-collision-distance`
: If this tie is closer than this amount to the previous tie in the set being considered, an increasingly large tie-tie collision penalty is incurred.

`tie-tie-collision-penalty`
: Demerit factor for a tie in the set being considered which is close to the previous one.

`horizontal-distance-penalty-factor`
: Demerit factor for ties in the set being considered which are horizontally distant from the note heads.

`vertical-distance-penalty-factor`
: Demerit factor for ties in the set being considered which are vertically distant from the note heads.

`same-dir-as-stem-penalty`
: Demerit if tie is on the same side as a stem or on the opposite side to the one specified.

`intra-space-threshold`
: If the tie's height (in half staff-spaces) is less than this it is positioned between two adjacent staff lines; otherwise it is positioned to straddle a staff line further from the note heads.

`outer-tie-length-symmetry-penalty-factor`
: Demerit factor for ties horizontally positioned unsymmetrically with respect to the two note heads.

`outer-tie-vertical-distance-symmetry-penalty-factor`
: Demerit factor for ties vertically positioned unsymmetrically with respect to the two note heads.

`outer-tie-vertical-gap`
: Amount (in half staff-spaces) by which a tie is moved away from the note heads if it is closer to either of them than 0.25 half staff-spaces.

`skyline-padding`
: Padding of the skylines around note heads in chords.

`single-tie-region-size`
: The number of candidate ties to generate when only a single tie is required. Successive candidates differ in their initial vertical position by half a staff-space.

`multi-tie-region-size`
: The number of variations that are tried for the extremal ties in a chord. Variations differ in their initial vertical position by half a staff-space.

User settable properties:

`annotation` (string)
: Annotate a grob for debug purposes.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current

staff-line thickness (i.e. the visual output is influenced by changes to *Staff*.StaffSymbol.thickness).

This grob interface is used in the following graphical object(s): Section 3.1.124 [Tie], page 513.

3.2.129 time-signature-interface

A time signature, in different styles. The following values for `style` are are recognized:

`C` 4/4 and 2/2 are typeset as C and struck C, respectively. All other time signatures are written with two digits. The value `default` is equivalent to `C`.

`neomensural`
2/2, 3/2, 2/4, 3/4, 4/4, 6/4, 9/4, 4/8, 6/8, and 9/8 are typeset with neo-mensural style mensuration marks. All other time signatures are written with two digits.

`mensural` 2/2, 3/2, 2/4, 3/4, 4/4, 6/4, 9/4, 4/8, 6/8, and 9/8 are typeset with mensural style mensuration marks. All other time signatures are written with two digits.

`single-digit`
All time signatures are typeset with a single digit, e.g., 3/2 is written as 3.

`numbered` All time signatures are typeset with two digits.

User settable properties:

`fraction` (fraction, as pair)
Numerator and denominator of a time signature object.

`style` (symbol)
This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

This grob interface is used in the following graphical object(s): Section 3.1.126 [TimeSignature], page 515.

3.2.130 trill-pitch-accidental-interface

An accidental for trill pitch.

This grob interface is used in the following graphical object(s): Section 3.1.127 [TrillPitchAccidental], page 518.

3.2.131 trill-spanner-interface

A trill spanner.

This grob interface is used in the following graphical object(s): Section 3.1.130 [TrillSpanner], page 521.

3.2.132 tuplet-bracket-interface

A bracket with a number in the middle, used for tuplets. When the bracket spans a line break, the value of `break-overshoot` determines how far it extends beyond the staff. At a line break, the markups in the `edge-text` are printed at the edges.

User settable properties:

`avoid-scripts` (boolean)
If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
: This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: `(`*`left-text . right-text`*`)`.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`positions` (pair of numbers)
: Pair of staff coordinates `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(`*left* `.` *right*`)`, where both *left* and *right* are in `staff-space` units of the current staff.

Internal properties:

`note-columns` (array of grobs)
: An array of `NoteColumn` grobs.

`scripts` (array of grobs)
: An array of `Script` objects.

`tuplet-number` (graphical (layout) object)
: The number for a bracket.

`tuplets` (array of grobs)
: An array of smaller tuplet brackets.

This grob interface is used in the following graphical object(s): Section 3.1.64 [Ligature-Bracket], page 449, and Section 3.1.131 [TupletBracket], page 522.

3.2.133 tuplet-number-interface

The number for a bracket.

User settable properties:

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`knee-to-beam` (boolean)
: Determines whether a tuplet number will be positioned next to a kneed beam.

Internal properties:

`bracket` (graphical (layout) object)
: The bracket for a number.

This grob interface is used in the following graphical object(s): Section 3.1.132 [TupletNumber], page 524.

3.2.134 unbreakable-spanner-interface

A spanner that should not be broken across line breaks. Override with `breakable=##t`.

User settable properties:

`breakable` (boolean)
: Allow breaks here.

This grob interface is used in the following graphical object(s): Section 3.1.19 [Beam], page 390, and Section 3.1.48 [Glissando], page 430.

3.2.135 vaticana-ligature-interface

A vaticana style Gregorian ligature.

User settable properties:

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff*`.StaffSymbol.thickness`).

Internal properties:

`add-cauda` (boolean)
: Does this flexa require an additional cauda on the left side?

`add-join` (boolean)
: Is this ligature head-joined with the next one by a vertical line?

`add-stem` (boolean)
: Is this ligature head a virga and therefore needs an additional stem on the right side?

`delta-position` (number)
: The vertical position difference.

`flexa-height` (dimension, in staff space)
: The height of a flexa shape in a ligature grob (in `staff-space` units).

`flexa-width` (dimension, in staff space)
: The width of a flexa shape in a ligature grob in (in `staff-space` units).

`x-offset` (dimension, in staff space)
: Extra horizontal offset for ligature heads.

This grob interface is used in the following graphical object(s): Section 3.1.80 [NoteHead], page 467, and Section 3.1.135 [VaticanaLigature], page 527.

3.2.136 volta-bracket-interface

Volta bracket with number.

User settable properties:

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

Internal properties:

`bars` (array of grobs)
: An array of bar line pointers.

This grob interface is used in the following graphical object(s): Section 3.1.139 [VoltaBracket], page 531.

3.2.137 volta-interface

A volta repeat.

This grob interface is used in the following graphical object(s): Section 3.1.139 [VoltaBracket], page 531, and Section 3.1.140 [VoltaBracketSpanner], page 532.

3.3 User backend properties

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)
: Alteration numbers for accidental.

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`annotation` (string)
: Annotate a grob for debug purposes.

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)
: Arrow length.

`arrow-width` (number)
: Arrow width.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

`automatically-numbered` (boolean)
: Should a footnote be automatically numbered?

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`avoid-scripts` (boolean)
: If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`axes` (list)
: List of axis numbers. In the case of alignment grobs, this should contain only one number.

`bar-extent` (pair of numbers)
: The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`base-shortest-duration` (moment)
: Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`baseline-skip` (dimension, in staff space)
: Distance between base lines of multiple lines of text.

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
: Width of the tremolo sign.

`beamed-stem-shorten` (list)
: How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
: A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
: The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`before-line-breaking` (boolean)
: Dummy property, used to trigger a callback function.

`between-cols` (pair)
: Where to attach a loose column to.

`bound-details` (list)
: An alist of properties for determining attachments of spanners to edges.

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
: This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)
: Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)
: This is a vector of 3 lists: `#(`*`end-of-line unbroken start-of-line`*`)`. Each list contains *break-align symbols* that specify an order of breakable items (see Section "break-alignment-interface" in *Internals Reference*).

  For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
```

```
ambitus
breathing-sign
time-signature
clef
cue-clef
staff-bar
key-cancellation
key-signature
custos))
```

`break-align-symbol` (symbol)
: This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" in *Internals Reference*.

`break-align-symbols` (list)
: A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" in *Internals Reference*.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`breakable` (boolean)
: Allow breaks here.

`broken-bound-padding` (number)
: The amount of padding to insert when a spanner is broken at a line break.

`chord-dots-limit` (integer)
: Limits the column of dots on each chord to the height of the chord plus `chord-dots-limit` staff-positions.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`clef-alignments` (list)
: An alist of parent-alignments that should be used for clef modifiers with various clefs

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`color` (color)
: The color of this grob.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`count-from` (integer)
: The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`damping` (number)
: Amount of beam slope damping.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`dash-fraction` (number)
: Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`default-direction` (direction)
: Direction determined by note head positions.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`digit-names` (vector)
: Names for string finger digits.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`dot-count` (integer)
: The number of dots.

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)
: List consisting of (*`description string-number fret-number finger-number`*) entries used to define fret diagrams.

`double-stem-separation` (number)
: The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: (*`left-height . right-height`*).

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: (*`left-text . right-text`*).

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`extra-dy` (number)
: Slope glissandi this much extra.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flag-count` (number)
: The number of tremolo beams.

`flag-style` (symbol)
: The style of the flag to be used with `MetronomeMark`. Available are `'modern-straight-flag`, `'old-straight-flag`, `flat-flag`, `mensural` and `'default`

`flat-positions` (list)
: Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the

staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-features` (list)
: Opentype features.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`footnote` (boolean)
: Should this be a footnote or in-note?

`footnote-music` (music)
: Music creating a footnote.

`footnote-text` (markup)
: A footnote for the grob.

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Section "note-collision-interface" in *Internals Reference*.

`forced-spacing` (number)
: Spacing forced between grobs, used in various ligature engravers.

`fraction` (fraction, as pair)
: Numerator and denominator of a time signature object.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a (*property* . *value*) pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.

- `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
- `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
- `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
- `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
- `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
- `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
- `fret-count` – The number of frets. Default 4.
- `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
- `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
- `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
- `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
- `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
- `paren-padding` – The padding for the parenthesis. Default 0.05.
- `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
- `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
- `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.

- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`full-size-change` (boolean)
: Don't make a change clef smaller.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`grow-direction` (direction)
: Crescendo or decrescendo?

`hair-thickness` (number)
: Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a `(property . value)` pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.

- `box-height` – Height of each pedal box. Default value 1.0.
- `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
- `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
- `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
- `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
- `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizon-padding` (number)
: The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Section "note-collision-interface" in *Internals Reference*.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`id` (string)
: An id string for the grob.

`ignore-ambitus` (boolean)
: If set, don't consider this notehead for ambitus calculation.

`ignore-collision` (boolean)
: If set, don't do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-index` (integer)
: If debugging is set, set beam and slur configuration to this index, and print the respective scores.

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`knee` (boolean)
: Is this beam kneed?

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`knee-to-beam` (boolean)
: Determines whether a tuplet number will be positioned next to a kneed beam.

`labels` (list)
: List of labels (symbols) placed on a column.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-extra` (dimension, in staff space)
: Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)
: Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`ledger-positions-function` (any type)
: A quoted Scheme procedure that takes a `StaffSymbol` grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual "pen" that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`long-text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`max-symbol-separation` (number)
: The maximum distance between symbols making up a church rest.

`maximum-gap` (number)
: Maximum value allowed for `gap` property.

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Section "note-collision-interface" in *Internals Reference*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-after-break` (dimension, in staff space)
: If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance to the notehead.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

`non-default` (boolean)
: Set for manually specified clefs and keys.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`note-collision-threshold` (dimension, in staff space)
: Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`number-type` (symbol)
: Numbering style. Choices include `roman-lower`, `roman-upper` and `arabic`.

`output-attributes` (list)
: An alist of attributes for the grob, to be included in output files. When the SVG typesetting backend is used, the attributes are assigned to a group (`<g>`) containing all of the stencils that comprise a given grob. For example, `'((id . 123) (class . foo) (data-whatever . \bar"))` will produce `<g id=\123" class=\foo" data-whatever=\bar"> ... </g>`. In the Postscript backend, where there is no way to group items, the setting of the output-attributes property will have no effect.

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-placement-directive` (symbol)
: One of four directives telling how outside staff objects should be placed.
  - `left-to-right-greedy` – Place each successive grob from left to right.
  - `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
  - `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
  - `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping (*name* . *name*) to distances.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`parent-alignment-X` (number)
: Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`parent-alignment-Y` (number)
: Like `parent-alignment-X` but for the Y axis.

`parenthesis-friends` (list)
: A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates (*left* . *right*), where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`protrusion` (number)
: In an arpeggio bracket, the length of the horizontal edges.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`remove-layer` (index or symbol)
: When set as a positive integer, the `Keep_alive_together_engraver` removes all `VerticalAxisGroup` grobs with a `remove-layer` larger than the smallest retained

`remove-layer`. Set to `#f` to make a layer independent of the `Keep_alive_together_engraver`. Set to `'()`, the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: `#'any` to keep the layer alive with any other layer in the group; `#'above` or `#'below` to keep the layer alive with the context immediately before or after it, respectively.

`replacement-alist` (list)
: Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

`restore-first` (boolean)
: Print a natural before the accidental.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `'(45 0 0)` rotates by 45 degrees around the center of this object.

`round-up-exceptions` (list)
: A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)
: Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a 3/2 measure.

`rounded` (boolean)
: Decide whether lines should be drawn rounded or not.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`segno-kern` (number)
: The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`shape` (symbol)
: This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

`sharp-positions` (list)
: Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`shortest-duration-space` (number)
: Start with this multiple of `spacing-increment` space for the shortest duration. See also Section "spacing-spanner-interface" in *Internals Reference*.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`side-axis` (number)
: If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`simple-Y` (boolean)
: Should the Y placement of a spanner disregard changes in system heights?

`size` (number)
: The ratio of the size of the object to its default size.

`skip-quanting` (boolean)
: Should beam quanting be skipped?

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`snap-radius` (number)
: The maximum distance between two objects that will cause them to snap to alignment along an axis.

`space-alist` (list)
: An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section "break-alignment-interface" in *Internals Reference*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`spacing-increment` (dimension, in staff space)
: The unit of length for note-spacing. Typically, the width of a note head. See also Section "spacing-spanner-interface" in *Internals Reference*.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

`spanner-id` (index or symbol)
: An identifier to distinguish concurrent spanners.

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stacking-dir` (direction)
: Stack objects in which direction?

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.

- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`stem-begin-position` (number)
: User override for the begin position of a stem.

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

`stemlet-length` (number)
: How long should be a stem over a rest?

`stencil` (stencil)
: The symbol to print.

`stencils` (list)
: Multiple stencils, used as intermediate value.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`text` (markup)
: Text markup. See Section "Formatting text" in *Notation Reference*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`thick-thickness` (number)
: Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`tie-configuration` (list)
: List of `(position . dir)` pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`toward-stem-shift-in-column` (number)
: Amount by which a script is shifted toward the stem if its direction coincides with the stem direction and it is associated with a `ScriptColumn` object. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

`usable-duration-logs` (list)
: List of `duration-log`s that can be used in typesetting the grob.

`use-skylines` (boolean)
: Should skylines be used for side positioning?

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`voiced-position` (number)
: The staff-position of a voiced `Rest`, negative if the rest has `direction DOWN`.

`when` (moment)
: Global time step associated with this column.

`whiteout` (boolean-or-number)
: If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white

background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`. Usually `#f` by default.

`whiteout-style` (symbol)
: Determines the shape of the `whiteout` background. Available are `'outline`, `'rounded-box`, and the default `'box`. There is one exception: Use `'special` for `LyricHyphen`.

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

`X-align-on-main-noteheads` (boolean)
: If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

`zigzag-length` (dimension, in staff space)
: The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

3.4 Internal backend properties

`accidental-grob` (graphical (layout) object)
: The accidental for this note.

`accidental-grobs` (list)
: An alist with `(`*`notename . groblist`*`)` entries.

`add-cauda` (boolean)
: Does this flexa require an additional cauda on the left side?

`add-join` (boolean)
: Is this ligature head-joined with the next one by a vertical line?

`add-stem` (boolean)
: Is this ligature head a virga and therefore needs an additional stem on the right side?

`adjacent-pure-heights` (pair)
: A pair of vectors. Used by a `VerticalAxisGroup` to cache the `Y-extent`s of different column ranges.

`adjacent-spanners` (array of grobs)
: An array of directly neighboring dynamic spanners.

`all-elements` (array of grobs)
: An array of all grobs in this line. Its function is to protect objects from being garbage collected.

`ascendens` (boolean)
: Is this neume of ascending type?

`auctum` (boolean)
: Is this neume liquescentically augmented?

`axis-group-parent-X` (graphical (layout) object)
: Containing X axis group.

`axis-group-parent-Y` (graphical (layout) object)
: Containing Y axis group.

`bars` (array of grobs)
: An array of bar line pointers.

`beam` (graphical (layout) object)
: A pointer to the beam, if applicable.

`beam-segments` (list)
: Internal representation of beam segments.

`begin-of-line-visible` (boolean)
: Set to make `ChordName` or `FretBoard` be visible only at beginning of line or at chord changes.

`bound-alignment-interfaces` (list)
: Interfaces to be used for positioning elements that align with a column.

`bounded-by-me` (array of grobs)
: An array of spanners that have this column as start/begin point. Only columns that have grobs or act as bounds are spaced.

`bracket` (graphical (layout) object)
: The bracket for a number.

`bracket-text` (graphical (layout) object)
: The text for an analysis bracket.

`c0-position` (integer)
: An integer indicating the position of middle C.

`cause` (any type)
: Any kind of causation objects (i.e., music, or perhaps translator) that was the cause for this grob.

`cavum` (boolean)
: Is this neume outlined?

`columns` (array of grobs)
: An array of grobs, typically containing `PaperColumn` or `NoteColumn` objects.

`concurrent-hairpins` (array of grobs)
: All concurrent hairpins.

`conditional-elements` (array of grobs)
: Internal use only.

`context-info` (integer)
: Within a ligature, the final glyph or shape of a head may be affected by the left and/or right neighbour head. `context-info` holds for each head such information about the left and right neighbour, encoded as a bit mask.

`covered-grobs` (array of grobs)
: Grobs that could potentially collide with a beam.

`cross-staff` (boolean)
: True for grobs whose `Y-extent` depends on inter-staff spacing. The extent is measured relative to the grobs's parent staff (more generally, its `VerticalAxisGroup`) so this boolean flags grobs that are not rigidly fixed to their parent staff. Beams that join notes from two staves are `cross-staff`. Grobs that are positioned around such beams are also `cross-staff`. Grobs that are grouping objects, however, like `VerticalAxisGroups` will not in general be marked `cross-staff` when some of the members of the group are `cross-staff`.

`delta-position` (number)
: The vertical position difference.

`deminutum` (boolean)
: Is this neume deminished?

`descendens` (boolean)
: Is this neume of descendent type?

`direction-source` (graphical (layout) object)
: In case `side-relative-direction` is set, which grob to get the direction from.

`display-cautionary` (boolean)
: Should the grob be displayed as a cautionary grob?

`dot` (graphical (layout) object)
: A reference to a `Dots` object.

`dots` (array of grobs)
: Multiple `Dots` objects.

`elements` (array of grobs)
: An array of grobs; the type is depending on the grob where this is set in.

`encompass-objects` (array of grobs)
: Objects that a slur should avoid in addition to notes and stems.

`figures` (array of grobs)
: Figured bass objects for continuation line.

`flag` (graphical (layout) object)
: A pointer to a `Flag` object.

`flexa-height` (dimension, in staff space)
: The height of a flexa shape in a ligature grob (in `staff-space` units).

`flexa-interval` (integer)
: The interval spanned by the two notes of a flexa shape (1 is a second, 7 is an octave).

`flexa-width` (dimension, in staff space)
: The width of a flexa shape in a ligature grob in (in `staff-space` units).

`font` (font metric)
: A cached font metric object.

`footnote-stencil` (stencil)
: The stencil of a system's footnotes.

`footnotes-after-line-breaking` (array of grobs)
: Footnote grobs of a broken system.

`footnotes-before-line-breaking` (array of grobs)
: Footnote grobs of a whole system.

`forced` (boolean)
: Manually forced accidental.

`glissando-index` (integer)
: The index of a glissando in its note column.

`grace-spacing` (graphical (layout) object)
: A run of grace notes.

`has-span-bar` (pair)
: A pair of grobs containing the span bars to be drawn below and above the staff. If no span bar is in a position, the respective element is set to `#f`.

`head-width` (dimension, in staff space)
: The width of this ligature head.

`heads` (array of grobs)
: An array of note heads.

`ideal-distances` (list)
: (*obj* . (*dist* . *strength*)) pairs.

`important-column-ranks` (vector)
: A cache of columns that contain `items-worth-living` data.

`in-note-direction` (direction)
: Direction to place in-notes above a system.

`in-note-padding` (number)
: Padding between in-notes.

`in-note-stencil` (stencil)
: The stencil of a system's in-notes.

`inclinatum` (boolean)
: Is this neume an inclinatum?

`interfaces` (list)
: A list of symbols indicating the interfaces supported by this object. It is initialized from the `meta` field.

`items-worth-living` (array of grobs)
: An array of interesting items. If empty in a particular staff, then that staff is erased.

`keep-alive-with` (array of grobs)
: An array of other `VerticalAxisGroup`s. If any of them are alive, then we will stay alive.

`least-squares-dy` (number)
: The ideal beam slope, without damping.

`left-items` (array of grobs)
: Grobs organized on the left by a spacing object.

`left-neighbor` (graphical (layout) object)
: The right-most column that has a spacing-wish for this column.

`ligature-flexa` (boolean)
: request joining note to the previous one in a flexa.

`linea` (boolean)
: Attach vertical lines to this neume?

`make-dead-when` (array of grobs)
: An array of other `VerticalAxisGroup`s. If any of them are alive, then we will turn dead.

`maybe-loose` (boolean)
: Used to mark a breakable column that is loose if and only if it is in the middle of a line.

`melody-spanner` (graphical (layout) object)
: The `MelodyItem` object for a stem.

`meta` (list) Provide meta information. It is an alist with the entries `name` and `interfaces`.

`minimum-distances` (list)
: A list of rods that have the format `(obj . dist)`.

`minimum-translations-alist` (list)
: An list of translations for a given start and end point.

`neighbors` (array of grobs)
: The X-axis neighbors of a grob. Used by the pure-from-neighbor-interface to determine various grob heights.

`normal-stems` (array of grobs)
: An array of visible stems.

`note-collision` (graphical (layout) object)
: The `NoteCollision` object of a dot column.

`note-columns` (array of grobs)
: An array of `NoteColumn` grobs.

`note-head` (graphical (layout) object)
: A single note head.

`note-heads` (array of grobs)
: An array of note head grobs.

`numbering-assertion-function` (any type)
: The function used to assert that footnotes are receiving correct automatic numbers.

`oriscus` (boolean)
: Is this neume an oriscus?

`pedal-text` (graphical (layout) object)
: A pointer to the text of a mixed-style piano pedal.

`pes-or-flexa` (boolean)
: Shall this neume be joined with the previous head?

`positioning-done` (boolean)
: Used to signal that a positioning element did its job. This ensures that a positioning is only done once.

`prefix-set` (number)
: A bit mask that holds all Gregorian head prefixes, such as `\virga` or `\quilisma`.

`primitive` (integer)
: A pointer to a ligature primitive, i.e., an item similar to a note head that is part of a ligature.

`pure-relevant-grobs` (array of grobs)
: All the grobs (items and spanners) that are relevant for finding the `pure-Y-extent`

`pure-relevant-items` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-relevant-spanners` (array of grobs)
: A subset of elements that are relevant for finding the `pure-Y-extent`.

`pure-Y-common` (graphical (layout) object)
: A cache of the `common_refpoint_of_array` of the `elements` grob set.

`pure-Y-extent` (pair of numbers)
: The estimated height of a system.

`pure-Y-offset-in-progress` (boolean)
: A debugging aid for catching cyclic dependencies.

`quantize-position` (boolean)
: If set, a vertical alignment is aligned to be within staff spaces.

`quantized-positions` (pair of numbers)
: The beam positions after quanting.

`quilisma` (boolean)
: Is this neume a quilisma?

`rest` (graphical (layout) object)
: A pointer to a `Rest` object.

`rest-collision` (graphical (layout) object)
: A rest collision that a rest is in.

`rests` (array of grobs)
: An array of rest objects.

`right-items` (array of grobs)
: Grobs organized on the right by a spacing object.

`right-neighbor` (graphical (layout) object)
: See `left-neighbor`.

`script-column` (graphical (layout) object)
: A `ScriptColumn` associated with a `Script` object.

`script-stencil` (pair)
: A pair `(`*`type . arg`*`)` which acts as an index for looking up a `Stencil` object.

`scripts` (array of grobs)
: An array of `Script` objects.

`shorten` (dimension, in staff space)
: The amount of space that a stem is shortened. Internally used to distribute beam shortening over stems.

`side-support-elements` (array of grobs)
: The side support, an array of grobs.

`slur` (graphical (layout) object)
: A pointer to a `Slur` object.

`space-increment` (dimension, in staff space)
: The amount by which the total duration of a multimeasure rest affects horizontal spacing. Each doubling of the duration adds `space-increment` to the length of the bar.

`spacing` (graphical (layout) object)
: The spacing spanner governing this section.

`spacing-wishes` (array of grobs)
: An array of note spacing or staff spacing objects.

`span-start` (boolean)
: Is the note head at the start of a spanner?

`spanner-broken` (boolean)
: Indicates whether spanner alignment should be broken after the current spanner.

`spanner-placement` (direction)
: The place of an annotation on a spanner. `LEFT` is for the first spanner, and `RIGHT` is for the last. `CENTER` will place it on the broken spanner that falls closest to the center of the length of the entire spanner, although this behavior is unpredictable in situations with lots of rhythmic diversity. For predictable results, use `LEFT` and `RIGHT`.

`staff-grouper` (graphical (layout) object)
: The staff grouper we belong to.

`staff-symbol` (graphical (layout) object)
: The staff symbol grob that we are in.

`stem` (graphical (layout) object)
: A pointer to a `Stem` object.

`stem-info` (pair)
: A cache of stem parameters.

`stems` (array of grobs)
: An array of stem objects.

`stropha` (boolean)
: Is this neume a stropha?

`system-Y-offset` (number)
: The Y-offset (relative to the bottom of the top-margin of the page) of the system to which this staff belongs.

`tie` (graphical (layout) object)
: A pointer to a `Tie` object.

`ties` (array of grobs)
: A grob array of `Tie` objects.

`tremolo-flag` (graphical (layout) object)
: The tremolo object on a stem.

`tuplet-number` (graphical (layout) object)
: The number for a bracket.

`tuplet-start` (boolean)
: Is stem at the start of a tuplet?

`tuplets` (array of grobs)
: An array of smaller tuplet brackets.

`vertical-alignment` (graphical (layout) object)
: The VerticalAlignment in a System.

`vertical-skyline-elements` (array of grobs)
: An array of grobs used to create vertical skylines.

`virga` (boolean)
: Is this neume a virga?

`X-common` (graphical (layout) object)
: Common reference point for axis group.

`x-offset` (dimension, in staff space)
: Extra horizontal offset for ligature heads.

`Y-common` (graphical (layout) object)
: See `X-common`.

4 Scheme functions

`ly:add-context-mod` *contextmods modification* [Function]
Adds the given context *modification* to the list *contextmods* of context modifications.

`ly:add-file-name-alist` *alist* [Function]
Add mappings for error messages from *alist*.

`ly:add-interface` *iface desc props* [Function]
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *callback disp cl* [Function]
Add the single-argument procedure *callback* as listener to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it calls *callback* with it.

`ly:add-option` *sym val description* [Function]
Add a program option *sym*. *val* is the default value and *description* is a string description.

`ly:all-grob-interfaces` [Function]
Return the hash table with all grob interface descriptions.

`ly:all-options` [Function]
Get all option settings in an alist.

`ly:all-stencil-expressions` [Function]
Return all symbols recognized as stencil expressions.

`ly:angle` *x y* [Function]
Calculates angle in degrees of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:assoc-get` *key alist default-value strict-checking* [Function]
Return value if *key* in *alist*, else *default-value* (or `#f` if not specified). If *strict-checking* is set to `#t` and *key* is not in *alist*, a programming_error is output.

`ly:axis-group-interface::add-element` *grob grob-element* [Function]
Set *grob* the parent of *grob-element* on all axes of *grob*.

`ly:basic-progress` *str rest* [Function]
A Scheme callable function to issue a basic progress message *str*. The message is formatted with `format` and *rest*.

`ly:beam-score-count` [Function]
count number of beam scores.

`ly:bigpdfs` [Function]
Return true if the command line includes the `--bigpdf` parameter.

`ly:book?` *x* [Function]
Is *x* a `Book` object?

`ly:book-add-bookpart!` *book-smob book-part* [Function]
Add *book-part* to *book-smob* book part list.

`ly:book-add-score!` *book-smob score* [Function]
Add *score* to *book-smob* score list.

`ly:book-book-parts` *book* [Function]
Return book parts in *book*.

`ly:book-header` *book* [Function]
Return header in *book*.

`ly:book-paper` *book* [Function]
Return paper in *book*.

`ly:book-process` *book-smob default-paper default-layout output* [Function]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-process-to-systems` *book-smob default-paper default-layout output* [Function]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-scores` *book* [Function]
Return scores in *book*.

`ly:book-set-header!` *book module* [Function]
Set the book header.

`ly:box?` *x* [Function]
Is *x* a `Box` object?

`ly:bp` *num* [Function]
num bigpoints (1/72th inch).

`ly:bracket` *a iv t p* [Function]
Make a bracket in direction *a*. The extent of the bracket is given by *iv*. The wings protrude by an amount of *p*, which may be negative. The thickness is given by *t*.

`ly:broadcast` *disp ev* [Function]
Send the stream event *ev* to the dispatcher *disp*.

`ly:camel-case->lisp-identifier` *name-sym* [Function]
Convert `FooBar_Bla` to `foo-bar-bla` style symbol.

`ly:chain-assoc-get` *key achain default-value strict-checking* [Function]
Return value for *key* from a list of alists *achain*. If no entry is found, return *default-value* or `#f` if *default-value* is not specified. With *strict-checking* set to `#t`, a programming_error is output in such cases.

`ly:check-expected-warnings` [Function]
Check whether all expected warnings have really been triggered.

`ly:cm` *num* [Function]
num cm.

`ly:command-line-code` [Function]
The Scheme code specified on command-line with `-e`.

`ly:command-line-options` [Function]
The Scheme options specified on command-line with `-d`.

`ly:connect-dispatchers` *to from* [Function]
Make the dispatcher *to* listen to events from *from*.

`ly:context?` *x* [Function]
Is *x* a `Context` object?

`ly:context-current-moment` *context* [Function]
Return the current moment of *context*.

`ly:context-def?` *x* [Function]
Is *x* a `Context_def` object?

`ly:context-def-lookup` *def sym val* [Function]
Return the value of *sym* in context definition *def* (e.g., `\Voice`). If no value is found, return *val* or `'()` if *val* is undefined. *sym* can be any of '`default-child`', '`consists`', '`description`', '`aliases`', '`accepts`', '`property-ops`', '`context-name`', '`group-type`'.

`ly:context-def-modify` *def mod* [Function]
Return the result of applying the context-mod *mod* to the context definition *def*. Does not change *def*.

`ly:context-event-source` *context* [Function]
Return `event-source` of context *context*.

`ly:context-events-below` *context* [Function]
Return a `stream-distributor` that distributes all events from *context* and all its subcontexts.

`ly:context-find` *context name* [Function]
Find a parent of *context* that has name or alias *name*. Return `#f` if not found.

`ly:context-grob-definition` *context name* [Function]
Return the definition of *name* (a symbol) within *context* as an alist.

`ly:context-id` *context* [Function]
Return the ID string of *context*, i.e., for `\context Voice = "one" ...` return the string `one`.

`ly:context-matched-pop-property` *context grob cell* [Function]
This undoes a particular `\override`, `\once \override` or `\once \revert` when given the specific alist pair to undo.

`ly:context-mod?` *x* [Function]
Is *x* a `Context_mod` object?

`ly:context-mod-apply!` *context mod* [Function]
Apply the context modification *mod* to *context*.

`ly:context-name` *context* [Function]
Return the name of *context*, i.e., for `\context Voice = "one" ...` return the symbol `Voice`.

`ly:context-now` *context* [Function]
Return `now-moment` of context *context*.

`ly:context-parent` *context* [Function]
Return the parent of *context*, `#f` if none.

`ly:context-property` *context sym def* [Function]
Return the value for property *sym* in *context*. If *def* is given, and property value is `'()`, return *def*.

`ly:context-property-where-defined` *context name* [Function]
Return the context above *context* where *name* is defined.

`ly:context-pushpop-property` *context grob eltprop val* [Function]
Do `\temporary \override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).

`ly:context-set-property!` *context name val* [Function]
Set value of property *name* in context *context* to *val*.

`ly:context-unset-property` *context name* [Function]
Unset value of property *name* in context *context*.

`ly:debug` *str rest* [Function]
A Scheme callable function to issue a debug message *str*. The message is formatted with `format` and *rest*.

`ly:default-scale` [Function]
Get the global default scale.

`ly:dimension?` *d* [Function]
Return *d* as a number. Used to distinguish length variables from normal numbers.

`ly:dir?` *s* [Function]
Is *s* a direction? Valid directions are `-1`, `0`, or `1`, where `-1` represents left or down, `1` represents right or up, and `0` represents a neutral direction.

`ly:directed` *direction magnitude* [Function]
Calculates an `(x . y)` pair with optional *magnitude* (defaulting to `1.0`) and *direction* specified either as an angle in degrees or a coordinate pair giving the direction. If *magnitude* is a pair, the respective coordinates are scaled independently, useful for ellipse drawings.

`ly:disconnect-dispatchers` *to from* [Function]
Stop the dispatcher *to* listening to events from *from*.

`ly:dispatcher?` *x* [Function]
Is *x* a `Dispatcher` object?

`ly:duration?` *x* [Function]
Is *x* a `Duration` object?

`ly:duration<?` *p1 p2* [Function]
Is *p1* shorter than *p2*?

`ly:duration->string` *dur* [Function]
Convert *dur* to a string.

`ly:duration-dot-count` *dur* [Function]
Extract the dot count from *dur*.

`ly:duration-factor` *dur* [Function]
Extract the compression factor from *dur*. Return it as a pair.

`ly:duration-length` *dur* [Function]
The length of the duration as a `moment`.

`ly:duration-log` *dur* [Function]
Extract the duration log from *dur*.

`ly:duration-scale` *dur* [Function]
Extract the compression factor from *dur*. Return it as a rational.

`ly:effective-prefix` [Function]
Return effective prefix.

`ly:encode-string-for-pdf` *str* [Function]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).

`ly:engraver-announce-end-grob` *engraver grob cause* [Function]
Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.

`ly:engraver-make-grob` *engraver grob-name cause* [Function]
Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.

`ly:error` *str rest* [Function]
A Scheme callable function to issue the error *str*. The error is formatted with `format` and *rest*.

`ly:event?` *obj* [Function]
Is *obj* a proper (non-rhythmic) event object?

`ly:event-deep-copy` *m* [Function]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym val* [Function]
Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or `'()` if *val* is not specified.

`ly:event-set-property!` *ev sym val* [Function]
Set property *sym* in event *ev* to *val*.

`ly:expand-environment` *str* [Function]
Expand `$VAR` and `${VAR}` in *str*.

`ly:expect-warning` *str rest* [Function]
A Scheme callable function to register a warning to be expected and subsequently suppressed. If the warning is not encountered, a warning about the missing warning will be shown. The message should be translated with `(_ ...)` and changing parameters given after the format string.

`ly:find-file` *name* [Function]
Return the absolute file name of *name*, or `#f` if not found.

`ly:font-config-add-directory` *dir* [Function]
Add directory *dir* to FontConfig.

`ly:font-config-add-font` *font* [Function]
Add font *font* to FontConfig.

`ly:font-config-display-fonts` [Function]
Dump a list of all fonts visible to FontConfig.

`ly:font-config-get-font-file` *name* [Function]
Get the file for font *name*.

`ly:font-design-size` *font* [Function]
Given the font metric *font*, return the design size, relative to the current output-scale.

`ly:font-file-name` *font* [Function]
Given the font metric *font*, return the corresponding file name.

`ly:font-get-glyph` *font name* [Function]
Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-charcode` *font name* [Function]
Return the character code for glyph *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-index` *font name* [Function]
Return the index for *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-index-to-charcode` *font index* [Function]
Return the character code for *index* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-magnification` *font* [Function]
Given the font metric *font*, return the magnification, relative to the current output-scale.

`ly:font-metric?` *x* [Function]
Is *x* a `Font_metric` object?

`ly:font-name` *font* [Function]
Given the font metric *font*, return the corresponding name.

`ly:font-sub-fonts` *font* [Function]
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

`ly:format` *str rest* [Function]
LilyPond specific format, supporting `~a` and `~[0-9]f`. Basic support for `~s` is also provided.

`ly:format-output` *context* [Function]
Given a global context in its final state, process it and return the `Music_output` object in its final state.

`ly:generic-bound-extent` *grob common* [Function]
Determine the extent of *grob* relative to *common* along the X axis, finding its extent as a bound when it a has `bound-alignment-interfaces` property list set and otherwise the full extent.

`ly:get-all-function-documentation` [Function]
Get a hash table with all LilyPond Scheme extension functions.

`ly:get-all-translators` [Function]
Return a list of all translator objects that may be instantiated.

`ly:get-cff-offset` *font-file-name idx* [Function]
Get the offset of 'CFF' table for *font_file_name*, returning it as an integer. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:get-context-mods` *contextmod* [Function]
Returns the list of context modifications stored in *contextmod*.

`ly:get-font-format` *font-file-name idx* [Function]
Get the font format for *font_file_name*, returning it as a symbol. The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:get-option` *var* [Function]
Get a global option setting.

`ly:get-spacing-spec` *from-scm to-scm* [Function]
Return the spacing spec going between the two given grobs, *from_scm* and *to_scm*.

`ly:get-undead` *undead* [Function]
Get back object from *undead*.

`ly:gettext` *original* [Function]
A Scheme wrapper function for `gettext`.

`ly:grob?` *x* [Function]
Is *x* a `Grob` object?

`ly:grob-alist-chain` *grob global* [Function]
Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, `font-defaults` from the layout block is taken.

`ly:grob-array?` *x* [Function]
Is *x* a `Grob_array` object?

`ly:grob-array->list` *grob-arr* [Function]
Return the elements of *grob-arr* as a Scheme list.

`ly:grob-array-length` *grob-arr* [Function]
Return the length of *grob-arr*.

`ly:grob-array-ref` *grob-arr index* [Function]
Retrieve the *index*th element of *grob-arr*.

`ly:grob-basic-properties` *grob* [Function]
Get the immutable properties of *grob*.

`ly:grob-chain-callback` *grob proc sym* [Function]
Find the callback that is stored as property *sym* of grob *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.

`ly:grob-common-refpoint` *grob other axis* [Function]
Find the common refpoint of *grob* and *other* for *axis*.

`ly:grob-common-refpoint-of-array` *grob others axis* [Function]
Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

`ly:grob-default-font` *grob* [Function]
Return the default font for grob *grob*.

`ly:grob-extent` *grob refp axis* [Function]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-get-vertical-axis-group-index` *grob* [Function]
Get the index of the vertical axis group the grob *grob* belongs to; return `-1` if none is found.

`ly:grob-interfaces` *grob* [Function]
Return the interfaces list of grob *grob*.

`ly:grob-layout` *grob* [Function]
Get `\layout` definition from grob *grob*.

`ly:grob-object` *grob sym* [Function]
Return the value of a pointer in grob *grob* of property *sym*. It returns `'()` (end-of-list) if *sym* is undefined in *grob*.

`ly:grob-original` *grob* [Function]
Return the unbroken original grob of *grob*.

`ly:grob-parent` *grob axis* [Function]
Get the parent of *grob*. *axis* is 0 for the X-axis, 1 for the Y-axis.

`ly:grob-pq<?` *a b* [Function]
Compare two grob priority queue entries. This is an internal function.

`ly:grob-properties` *grob* [Function]
Get the mutable properties of *grob*.

`ly:grob-properties?` *x* [Function]
Is *x* a `Grob_properties` object?

`ly:grob-property` *grob sym val* [Function]
Return the value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-property-data` *grob sym* [Function]
Return the value for property *sym* of *grob*, but do not process callbacks.

`ly:grob-pure-height` *grob refp beg end val* [Function]
Return the pure height of *grob* given refpoint *refp*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-pure-property` *grob sym beg end val* [Function]
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-relative-coordinate` *grob refp axis* [Function]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-robust-relative-extent` *grob refp axis* [Function]
Get the extent in *axis* direction of *grob* relative to the grob *refp*, or `(0,0)` if empty.

`ly:grob-script-priority-less` *a b* [Function]
Compare two grobs by script priority. For internal use.

`ly:grob-set-nested-property!` *grob symlist val* [Function]
Set nested property *symlist* in grob *grob* to value *val*.

`ly:grob-set-object!` *grob sym val* [Function]
Set *sym* in grob *grob* to value *val*.

`ly:grob-set-parent!` *grob axis parent-grob* [Function]
Set *parent-grob* the parent of grob *grob* in axis *axis*.

`ly:grob-set-property!` *grob sym val* [Function]
Set *sym* in grob *grob* to value *val*.

`ly:grob-spanned-rank-interval` *grob* [Function]
Returns a pair with the `rank` of the furthest left column and the `rank` of the furthest right column spanned by `grob`.

`ly:grob-staff-position` *sg* [Function]
Return the Y-position of *sg* relative to the staff.

`ly:grob-suicide!` *grob* [Function]
Kill *grob*.

`ly:grob-system` *grob* [Function]
Return the system grob of *grob*.

`ly:grob-translate-axis!` *grob d a* [Function]
Translate *grob* on axis *a* over distance *d*.

`ly:grob-vertical<?` *a b* [Function]
Does *a* lie above *b* on the page?

`ly:gulp-file` *name size* [Function]
Read *size* characters from the file *name*, and return its contents in a string. If *size* is undefined, the entire file is read. The file is looked up using the search path.

`ly:has-glyph-names?` *font-file-name idx* [Function]
Does the font for *font_file_name* have glyph names? The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:hash-table-keys` *tab* [Function]
Return a list of keys in *tab*.

`ly:inch` *num* [Function]
num inches.

`ly:input-both-locations` *sip* [Function]
Return input location in *sip* as `(file-name first-line first-column last-line last-column)`.

`ly:input-file-line-char-column` *sip* [Function]
Return input location in *sip* as `(file-name line char column)`.

`ly:input-location?` *x* [Function]
Is *x* a `Input` object?

`ly:input-message` *sip msg rest* [Function]
Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:input-warning` *sip msg rest* [Function]
Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:interpret-music-expression` *mus ctx* [Function]
Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.

`ly:interpret-stencil-expression` *expr func arg1 offset* [Function]
Parse *expr*, feed bits to *func* with first arg *arg1* having offset *offset*.

`ly:intlog2` *d* [Function]
The 2-logarithm of $1/d$.

`ly:item?` *g* [Function]
Is *g* an `Item` object?

`ly:item-break-dir` *it* [Function]
The break status direction of item *it*. `-1` means end of line, `0` unbroken, and `1` beginning of line.

`ly:item-get-column` *it* [Function]
Return the `PaperColumn` or `NonMusicalPaperColumn` associated with this `Item`.

`ly:iterator?` *x* [Function]
Is *x* a `Music_iterator` object?

`ly:length` *x y* [Function]
Calculates magnitude of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:lexer-keywords` *lexer* [Function]
Return a list of `(KEY . CODE)` pairs, signifying the LilyPond reserved words list.

`ly:lily-lexer?` *x* [Function]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Function]
Is *x* a `Lily_parser` object?

`ly:line-interface::line` *grob startx starty endx endy* [Function]
Make a line using layout information from grob *grob*.

`ly:listened-event-class?` *disp cl* [Function]
Does *disp* listen to any event type in the list *cl*?

`ly:listened-event-types` *disp* [Function]
Return a list of all event types that *disp* listens to.

`ly:listener?` *x* [Function]
Is *x* a `Listener` object?

`ly:make-book` *paper header scores* [Function]
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.

`ly:make-book-part` *scores* [Function]
Make a `\bookpart` containing `\scores`.

`ly:make-context-mod` *mod-list* [Function]
Creates a context modification, optionally initialized via the list of modifications *mod-list*.

`ly:make-dispatcher` [Function]
Return a newly created dispatcher.

`ly:make-duration` *length dotcount num den* [Function]
length is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.

The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

`ly:make-global-context` *output-def* [Function]
Set up a global interpretation context, using the output block *output-def*. The context is returned.

`ly:make-global-translator` *global* [Function]
Create a translator group and connect it to the global context *global*. The translator group is returned.

`ly:make-grob-properties` *alist* [Function]
This packages the given property list *alist* in a grob property container stored in a context property with the name of a grob.

`ly:make-moment` *m g gn gd* [Function]
Create the moment with rational main timing *m*, and optional grace timing *g*.

A *moment* is a point in musical time. It consists of a pair of rationals (*m*, *g*), where *m* is the timing for the main notes, and *g* the timing for grace notes. In absence of grace notes, *g* is zero.

For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.

`ly:make-music` *props* [Function]
Make a C++ `Music` object and initialize it with *props*.

This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.

`ly:make-music-function` *signature func* [Function]
Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*'s cdr is a list containing either `ly:music?` predicates or other type predicates. Its car is the syntax function to call.

`ly:make-music-relative!` *music pitch* [Function]
Make *music* relative to *pitch*, return final pitch.

`ly:make-output-def` [Function]
Make an output definition.

`ly:make-page-label-marker` *label* [Function]
Return page marker with label *label*.

`ly:make-page-permission-marker` *symbol permission* [Function]
Return page marker with page breaking and turning permissions.

`ly:make-pango-description-string` *chain size* [Function]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.

`ly:make-paper-outputter` *port format* [Function]
Create an outputter that evaluates within `output-`*format*, writing to *port*.

`ly:make-pitch` *octave note alter* [Function]
octave is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob` *type init rest* [Function]
Create a `Prob` object.

`ly:make-scale` *steps* [Function]
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

`ly:make-score` *music* [Function]
Return score with *music* encapsulated in it.

`ly:make-spring` *ideal min-dist* [Function]
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.

`ly:make-stencil` *expr xext yext* [Function]
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example `scm/output-ps.scm`.
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use `empty-interval` as its value), it is taken to be empty.

`ly:make-stream-event` *cl proplist* [Function]
Create a stream event of class *cl* with the given mutable property list.

`ly:make-undead` *object* [Function]
This packages *object* in a manner that keeps it from triggering "Parsed object should be dead" messages.

`ly:make-unpure-pure-container` *unpure pure* [Function]
Make an unpure-pure container. *unpure* should be an unpure expression, and *pure* should be a pure expression. If *pure* is omitted, the value of *unpure* will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.

`ly:message` *str rest* [Function]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.

`ly:minimal-breaking` *pb* [Function]
Break (pages and lines) the `Paper_book` object *pb* without looking for optimal spacing: stack as many lines on a page before moving to the next one.

`ly:mm` *num* [Function]
num mm.

`ly:module->alist` *mod* [Function]
Dump the contents of module *mod* as an alist.

`ly:module-copy` *dest src* [Function]
Copy all bindings from module *src* into *dest*.

`ly:modules-lookup` *modules sym def* [Function]
Look up *sym* in the list *modules*, returning the first occurence. If not found, return *def* or `#f` if *def* isn't specified.

`ly:moment?` *x* [Function]
Is *x* a `Moment` object?

`ly:moment<?` *a b* [Function]
Compare two moments.

`ly:moment-add` *a b* [Function]
Add two moments.

`ly:moment-div` *a b* [Function]
Divide two moments.

`ly:moment-grace` *mom* [Function]
Extract grace timing as a rational number from *mom*.

`ly:moment-grace-denominator` *mom* [Function]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Function]
Extract numerator from grace timing.

`ly:moment-main` *mom* [Function]
Extract main timing as a rational number from *mom*.

`ly:moment-main-denominator` *mom* [Function]
Extract denominator from main timing.

`ly:moment-main-numerator` *mom* [Function]
Extract numerator from main timing.

`ly:moment-mod` *a b* [Function]
Modulo of two moments.

`ly:moment-mul` *a b* [Function]
Multiply two moments.

`ly:moment-sub` *a b* [Function]
Subtract two moments.

`ly:music?` *obj* [Function]
Is *obj* a music object?

`ly:music-compress` *m factor* [Function]
Compress music object *m* by moment *factor*.

`ly:music-deep-copy` *m origin* [Function]
Copy *m* and all sub expressions of *m*. *m* may be an arbitrary type; cons cells and music are copied recursively. If *origin* is given, it is used as the origin for one level of music by calling `ly:set-origin!` on the copy.

`ly:music-duration-compress` *mus fact* [Function]
Compress *mus* by factor *fact*, which is a `Moment`.

`ly:music-duration-length` *mus* [Function]
Extract the duration field from *mus* and return the length.

`ly:music-function?` *x* [Function]
Is *x* a `Music_function` object?

`ly:music-function-extract` *x* [Function]
Return the Scheme function inside *x*.

`ly:music-function-signature` *x* [Function]
Return the function signature inside *x*.

`ly:music-length` *mus* [Function]
Get the length of music expression *mus* and return it as a `Moment` object.

`ly:music-list?` *lst* [Function]
Is *lst* a list of music objects?

`ly:music-mutable-properties` *mus* [Function]
Return an alist containing the mutable properties of *mus*. The immutable properties are not available, since they are constant and initialized by the `make-music` function.

`ly:music-output?` *x* [Function]
Is *x* a `Music_output` object?

`ly:music-property` *mus sym val* [Function]
Return the value for property *sym* of music expression *mus*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:music-set-property!` *mus sym val* [Function]
Set property *sym* in music expression *mus* to *val*.

`ly:music-transpose` *m p* [Function]
Transpose *m* such that central C is mapped to *p*. Return *m*.

`ly:note-column-accidentals` *note-column* [Function]
Return the `AccidentalPlacement` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-column-dot-column` *note-column* [Function]
Return the `DotColumn` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-head::stem-attachment` *font-metric glyph-name* [Function]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name*.

`ly:number->string` *s* [Function]
Convert *s* to a string without generating many decimals.

`ly:one-line-auto-height-breaking` *pb* [Function]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.

`ly:one-line-breaking` *pb* [Function]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.

`ly:one-page-breaking` *pb* [Function]
Put each score on a single page. The paper-height settings are modified so each score fits on one page, and the height of the page matches the height of the full score.

`ly:optimal-breaking` *pb* [Function]
Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in bother vertical and horizontal spacing.

`ly:option-usage` *port* [Function]
Print `ly:set-option` usage. Optional *port* argumentfor the destination defaults to current output port.

`ly:otf->cff` *otf-file-name idx* [Function]
Convert the contents of an OTF file to a CFF file, returning it as a string. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:otf-font?` *font* [Function]
Is *font* an OpenType font?

`ly:otf-font-glyph-info` *font glyph* [Function]
Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).

`ly:otf-font-table-data` *font tag* [Function]
Extract a table *tag* from *font*. Return empty string for non-existent *tag*.

`ly:otf-glyph-count` *font* [Function]
Return the number of glyphs in *font*.

`ly:otf-glyph-list` *font* [Function]
Return a list of glyph names for *font*.

`ly:output-def?` *x* [Function]
Is *x* a `Output_def` object?

`ly:output-def-clone` *def* [Function]
Clone output definition *def*.

`ly:output-def-lookup` *def sym val* [Function]
Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `'()` if *val* is undefined.

`ly:output-def-parent` *def* [Function]
Return the parent output definition of *def*.

`ly:output-def-scope` *def* [Function]
Return the variable scope inside *def*.

`ly:output-def-set-variable!` *def sym val* [Function]
Set an output definition *def* variable *sym* to *val*.

`ly:output-description` *output-def* [Function]
Return the description of translators in *output-def*.

`ly:output-find-context-def` *output-def context-name* [Function]
Return an alist of all context defs (matching *context-name*if given) in *output-def*.

`ly:output-formats` [Function]
Formats passed to `--format` as a list of strings, used for the output.

`ly:outputter-close` *outputter* [Function]
Close port of *outputter*.

`ly:outputter-dump-stencil` *outputter stencil* [Function]
Dump stencil *expr* onto *outputter*.

`ly:outputter-dump-string` *outputter str* [Function]
Dump *str* onto *outputter*.

`ly:outputter-module` *outputter* [Function]
Return output module of *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Function]
Eval *expr* in module of *outputter*.

`ly:outputter-port` *outputter* [Function]
Return output port for *outputter*.

`ly:page-marker?` *x* [Function]
Is *x* a `Page_marker` object?

`ly:page-turn-breaking` *pb* [Function]
Optimally break (pages and lines) the `Paper_book` object *pb* such that page turns only happen in specified places, returning its pages.

`ly:pango-font?` *f* [Function]
Is *f* a pango font?

`ly:pango-font-physical-fonts` *f* [Function]
Return alist of `(ps-name file-name font-index)` lists for Pango font *f*.

`ly:paper-book?` *x* [Function]
Is *x* a `Paper_book` object?

`ly:paper-book-header` *pb* [Function]
Return the header definition (`\header`) in `Paper_book` object *pb*.

`ly:paper-book-pages` *pb* [Function]
Return pages in `Paper_book` object *pb*.

`ly:paper-book-paper` *pb* [Function]
Return the paper output definition (`\paper`) in `Paper_book` object *pb*.

`ly:paper-book-performances` *pb* [Function]
Return performances in `Paper_book` object *pb*.

`ly:paper-book-scopes` *pb* [Function]
Return scopes in `Paper_book` object *pb*.

`ly:paper-book-systems` *pb* [Function]
Return systems in `Paper_book` object *pb*.

`ly:paper-column::break-align-width` *col align-syms* [Function]
Determine the extent along the X-axis of a grob used for break-alignment organized by column *col*. The grob is specified by *align-syms*, which contains either a single `break-align-symbol` or a list of such symbols.

`ly:paper-column::print` [Function]
Optional stencil for `PaperColumn` or`NonMusicalPaperColumn`. Draws the `rank` `number` of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.

`ly:paper-fonts` *def* [Function]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).

`ly:paper-get-font` *def chain* [Function]
Find a font metric in output definition *def* satisfying the font-qualifiers in alist chain *chain*, and return it. (An alist chain is a list of alists, containing grob properties.)

`ly:paper-get-number` *def sym* [Function]
Return the value of variable *sym* in output definition *def* as a double.

`ly:paper-outputscale` *def* [Function]
Return the output-scale for output definition *def*.

`ly:paper-score-paper-systems` *paper-score* [Function]
Return vector of `paper_system` objects from *paper-score*.

`ly:paper-system?` *obj* [Function]
Is *obj* a C++ `Prob` object of type `paper-system`?

`ly:paper-system-minimum-distance` *sys1 sys2* [Function]
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

`ly:parse-file` *name* [Function]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.

`ly:parse-string-expression` *parser-smob ly-code filename line* [Function]
Parse the string *ly-code* with *parser-smob*. Return the contained music expression. *filename* and *line* are optional source indicators.

`ly:parsed-undead-list!` [Function]
Return the list of objects that have been found live that should have been dead, and clear that list.

`ly:parser-clear-error` *parser* [Function]
Clear error flag for *parser*, defaulting to current parser.

`ly:parser-clone` *closures location* [Function]
Return a clone of current parser. An association list of port positions to closures can be specified in *closures* in order to have `$` and `#` interpreted in their original lexical environment. If *location* is a valid location, it becomes the source of all music expressions inside.

`ly:parser-define!` *symbol val* [Function]
Bind *symbol* to *val* in current parser's module.

`ly:parser-error` *msg input* [Function]
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.

`ly:parser-has-error?` *parser* [Function]
Does *parser* (defaulting to current parser) have an error flag?

`ly:parser-include-string` *ly-code* [Function]
Include the string *ly-code* into the input stream for current parser. Can only be used in immediate Scheme expressions (`$` instead of `#`).

`ly:parser-lexer` *parser* [Function]
Return the lexer for *parser*, defaulting to current parser

`ly:parser-lookup` *symbol* [Function]
Look up *symbol* in current parser's module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Function]
Return the base name of the output file. If `parser` is left off, use currently active parser.

`ly:parser-parse-string` *parser-smob ly-code* [Function]
Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.

`ly:parser-set-note-names` *names* [Function]
Replace current note names in parser. *names* is an alist of symbols. This only has effect if the current mode is notes.

`ly:performance-header` *performance* [Function]
Return header of performance.

`ly:performance-set-header!` *performance module* [Function]
Set the performance header.

`ly:performance-write` *performance filename name* [Function]
Write *performance* to *filename* storing *name* as the name of the performance in the file metadata.

`ly:pitch?` *x* [Function]
Is *x* a `Pitch` object?

`ly:pitch<?` *p1 p2* [Function]
Is *p1* lexicographically smaller than *p2*?

`ly:pitch-alteration` *pp* [Function]
Extract the alteration from pitch *pp*.

`ly:pitch-diff` *pitch root* [Function]
Return pitch *delta* such that *root* transposed by *delta* equals *pitch*.

`ly:pitch-negate` *p* [Function]
Negate *p*.

`ly:pitch-notename` *pp* [Function]
Extract the note name from pitch *pp*.

`ly:pitch-octave` *pp* [Function]
Extract the octave from pitch *pp*.

`ly:pitch-quartertones` *pp* [Function]
Calculate the number of quarter tones of *pp* from middle C.

`ly:pitch-semitones` *pp* [Function]
Calculate the number of semitones of *pp* from middle C.

`ly:pitch-steps` *p* [Function]
Number of steps counted from middle C of the pitch *p*.

`ly:pitch-tones` *pp* [Function]
Calculate the number of tones of *pp* from middle C as a rational number.

`ly:pitch-transpose` *p delta* [Function]
Transpose *p* by the amount *delta*, where *delta* is relative to middle C.

`ly:pointer-group-interface::add-grob` *grob sym grob-element* [Function]
Add *grob-element* to *grob*'s *sym* grob array.

`ly:position-on-line?` *sg spos* [Function]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).

`ly:prob?` *x* [Function]
Is *x* a `Prob` object?

`ly:prob-immutable-properties` *prob* [Function]
Retrieve an alist of immutable properties.

`ly:prob-mutable-properties` *prob* [Function]
Retrieve an alist of mutable properties.

`ly:prob-property` *prob sym val* [Function]
Return the value for property *sym* of Prob object *prob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:prob-property?` *obj sym* [Function]
Is boolean prop *sym* of *sym* set?

`ly:prob-set-property!` *obj sym value* [Function]
Set property *sym* of *obj* to *value*.

`ly:prob-type?` *obj type* [Function]
Is *obj* the specified prob-type?

`ly:programming-error` *str rest* [Function]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format` and *rest*.

`ly:progress` *str rest* [Function]
A Scheme callable function to print progress *str*. The message is formatted with `format` and *rest*.

`ly:property-lookup-stats` *sym* [Function]
Return hash table with a property access corresponding to *sym*. Choices are `prob`, `grob`, and `context`.

`ly:protects` [Function]
Return hash of protected objects.

`ly:pt` *num* [Function]
num printer points.

`ly:pure-call` *data grob start end rest* [Function]
Convert property *data* (unpure-pure container or procedure) to value in a pure context defined by *grob*, *start*, *end*, and possibly *rest* arguments.

`ly:register-stencil-expression` *symbol* [Function]
Add *symbol* as head of a stencil expression.

`ly:register-translator` *creator name description* [Function]
Register a translator *creator* (usually a descriptive alist or a function/closure returning one when given a context argument) with the given symbol *name* and the given *description* alist.

`ly:relative-group-extent` *elements common axis* [Function]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`ly:reset-all-fonts` [Function]
Forget all about previously loaded fonts.

`ly:round-filled-box` *xext yext blot* [Function]
Make a `Stencil` object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.

`ly:round-filled-polygon` *points blot extroversion* [Function]
Make a `Stencil` object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*. Optional*extroversion* shifts the outline outward, with thedefault of `-1.0` keeping the outer boundary ofthe outline just inside of the polygon.

`ly:run-translator` *mus output-def* [Function]
Process *mus* according to *output-def*. An interpretation context is set up, and *mus* is interpreted with it. The context is returned in its final state.

Optionally, this routine takes an object-key to to uniquely identify the score block containing it.

`ly:score?` *x* [Function]
Is *x* a `Score` object?

`ly:score-add-output-def!` *score def* [Function]
Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Function]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

`ly:score-error?` *score* [Function]
Was there an error in the score?

`ly:score-header` *score* [Function]
Return score header.

`ly:score-music` *score* [Function]
Return score music.

`ly:score-output-defs` *score* [Function]
All output definitions in a score.

`ly:score-set-header!` *score module* [Function]
Set the score header.

`ly:separation-item::print` [Function]
Optional stencil for `PaperColumn` or`NonMusicalPaperColumn`. Draws the `horizontal-skylines` of each `PaperColumn`, showing the shapes used to determine the minimum distances between `PaperColumns` at the note-spacing step, before staves have been spaced (vertically) on the page.

`ly:set-default-scale` *scale* [Function]
Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

`ly:set-grob-modification-callback` *cb* [Function]
Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property.

`ly:set-middle-C!` *context* [Function]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.

`ly:set-option` *var val* [Function]
Set a program option.

`ly:set-origin!` *m origin* [Function]
This sets the origin given in *origin* to *m*. *m* will typically be a music expression or a list of music. List structures are searched recursively, but recursion stops at the changed music expressions themselves. *origin* is generally of type `ly:input-location?`, defaulting to `(*location*)`. Other valid values for `origin` are a music expression which is then used as the source of location information, or `#f` or `'()` in which case no action is performed. The return value is *m* itself.

`ly:set-property-cache-callback` *cb* [Function]
Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property.

`ly:skyline?` *x* [Function]
Is *x* a `Skyline` object?

`ly:skyline-empty?` *sky* [Function]
Return whether *sky* is empty.

`ly:skyline-pair?` *x* [Function]
Is *x* a `Skyline_pair` object?

`ly:slur-score-count` [Function]
count number of slur scores.

`ly:smob-protects` [Function]
Return LilyPond's internal smob protection list.

`ly:solve-spring-rod-problem` *springs rods length ragged* [Function]
Solve a spring and rod problem for *count* objects, that are connected by *count*-1 *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs* argument has the format `(ideal, inverse_hook)` and *rods* is of the form `(idx1, idx2, distance)`.

length is a number, *ragged* a boolean.

The function returns a list containing the force (positive for stretching, negative for compressing and `#f` for non-satisfied constraints) followed by *spring-count*+1 positions of the objects.

`ly:source-file?` *x* [Function]
Is *x* a `Source_file` object?

`ly:source-files` *parser-smob* [Function]
A list of LilyPond files being processed;a PARSER may optionally be specified.

`ly:spanner?` *g* [Function]
Is *g* a spanner object?

`ly:spanner-bound` *spanner dir* [Function]
Get one of the bounds of *spanner*. *dir* is `-1` for left, and `1` for right.

`ly:spanner-broken-into` *spanner* [Function]
Return broken-into list for *spanner*.

`ly:spanner-set-bound!` *spanner dir item* [Function]
Set grob *item* as bound in direction *dir* for *spanner*.

`ly:spawn` *command rest* [Function]
Simple interface to g_spawn_sync *str*. The error is formatted with `format` and *rest*.

`ly:spring?` *x* [Function]
Is *x* a `Spring` object?

`ly:spring-set-inverse-compress-strength!` *spring strength* [Function]
Set the inverse compress *strength* of *spring*.

`ly:spring-set-inverse-stretch-strength!` *spring strength* [Function]
Set the inverse stretch *strength* of *spring*.

`ly:staff-symbol-line-thickness` *grob* [Function]
Returns the current staff-line thickness in the staff associated with *grob*, expressed as a multiple of the current staff-space height.

`ly:staff-symbol-staff-radius` *grob* [Function]
Returns the radius of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Function]
Returns the current staff-space height in the staff associated with *grob*, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`ly:start-environment` [Function]
Return the environment (a list of strings) that was in effect at program start.

`ly:stderr-redirect` *file-name mode* [Function]
Redirect stderr to *file-name*, opened with *mode*.

`ly:stencil?` *x* [Function]
Is *x* a `Stencil` object?

`ly:stencil-add` *args* [Function]
Combine stencils. Takes any number of arguments.

`ly:stencil-aligned-to` *stil axis dir* [Function]
Align *stil* using its own extents. *dir* is a number. `-1` and `1` are left and right, respectively. Other values are interpolated (so `0` means the center).

`ly:stencil-combine-at-edge` *first axis direction second padding* [Function]
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil axis* [Function]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.

`ly:stencil-expr` *stil* [Function]
Return the expression of *stil*.

`ly:stencil-extent` *stil axis* [Function]
Return a pair of numbers signifying the extent of *stil* in *axis* direction (`0` or `1` for x and y axis, respectively).

`ly:stencil-fonts` *s* [Function]
Analyze *s*, and return a list of fonts used in *s*.

`ly:stencil-in-color` *stc r g b* [Function]
Put *stc* in a different color.

`ly:stencil-outline` *stil outline* [Function]
Return a stencil with the stencil expression (inking) of stencil *stil* but with outline and dimensions from stencil *outline*.

`ly:stencil-rotate` *stil angle x y* [Function]
Return a stencil *stil* rotated *angle* degrees around the relative offset (*x*, *y*). E.g., an offset of (-1, 1) will rotate the stencil around the left upper corner.

`ly:stencil-rotate-absolute` *stil angle x y* [Function]
Return a stencil *stil* rotated *angle* degrees around point (*x*, *y*), given in absolute coordinates.

`ly:stencil-scale` *stil x y* [Function]
Scale stencil *stil* using the horizontal and vertical scaling factors *x* and *y*. Negative values will flip or mirror *stil* without changing its origin; this may result in collisions unless it is repositioned.

`ly:stencil-stack` *first axis direction second padding mindist* [Function]
Construct a stencil by stacking *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`. As opposed to `ly:stencil-combine-at-edge`, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.

If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, *padding* and *mindist* do not apply.

`ly:stencil-translate` *stil offset* [Function]
Return a *stil*, but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Function]
Return a copy of *stil* but translated by *amount* in *axis* direction.

`ly:stream-event?` *obj* [Function]
Is *obj* a `Stream_event` object?

`ly:string-percent-encode` *str* [Function]
Encode all characters in string *str* with hexadecimal percent escape sequences, with the following exceptions: characters `-`, `.`, `/`, and `_`; and characters in ranges `0-9`, `A-Z`, and `a-z`.

`ly:string-substitute` *a b s* [Function]
Replace string *a* by string *b* in string *s*.

`ly:system-font-load` *name* [Function]
Load the OpenType system font *name*`.otf`. Fonts loaded with this command must contain three additional SFNT font tables called `LILC`, `LILF`, and `LILY`, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.

Note that only `ly:font-get-glyph` and derived code (like `\lookup`) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.

`ly:text-interface::interpret-markup` [Function]
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.

layout is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.

`ly:translate-cpp-warning-scheme` *str* [Function]
Translates a string in C++ printf format and modifies it to use it for scheme formatting.

`ly:translator?` *x* [Function]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Function]
Return the context of the translator object *trans*.

`ly:translator-description` *creator* [Function]
Return an alist of properties of translator definition *creator*.

`ly:translator-group?` *x* [Function]
Is *x* a `Translator_group` object?

`ly:translator-name` *creator* [Function]
Return the type name of the translator definition *creator*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Function]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:truncate-list!` *lst i* [Function]
Take at most the first *i* of list *lst*.

`ly:ttf->pfa` *ttf-file-name idx* [Function]
Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:ttf-ps-name` *ttf-file-name idx* [Function]
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:type1->pfa` *type1-file-name* [Function]
Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in PFA format, pass through it.

`ly:undead?` *x* [Function]
Is *x* a `Undead` object?

`ly:unit` [Function]
Return the unit used for lengths as a string.

`ly:unpure-call` *data grob rest* [Function]
Convert property *data* (unpure-pure container or procedure) to value in an unpure context defined by *grob* and possibly *rest* arguments.

`ly:unpure-pure-container?` *x* [Function]
Is *x* a `Unpure_pure_container` object?

`ly:unpure-pure-container-pure-part` *pc* [Function]
Return the pure part of *pc*.

`ly:unpure-pure-container-unpure-part` *pc* [Function]
Return the unpure part of *pc*.

`ly:usage` [Function]
Print usage message.

`ly:verbose-output?` [Function]
Was verbose output requested, i.e. loglevel at least `DEBUG`?

`ly:version` [Function]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.

`ly:warning` *str rest* [Function]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.

`ly:warning-located` *location str rest* [Function]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with `format` and *rest*.

`ly:wide-char->utf-8` *wc* [Function]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.

Appendix A Indices

A.1 Concept index

(Index is nonexistent)

A.2 Function index