This manual is a technical reference for all internal elements used by LilyPond and all Scheme functions it provides. This information can be used to create tweaks and extensions, from simple output settings to advanced Scheme programming.

For more information about how this manual fits with the other documentation, or to read this manual in other formats, see Section “Manuals” in General Information. If you are missing any manuals, the complete documentation can be found at https://lilypond.org/.

Copyright © 2000–2022 by the authors

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections.

For LilyPond version 2.23.13
# Table of Contents

1 Music definitions .................................................. 1

1.1 Music expressions ................................................. 1

1.1.1 AbsoluteDynamicEvent ........................................ 1

1.1.2 AdHocJumpEvent .............................................. 1

1.1.3 AdHocMarkEvent .............................................. 1

1.1.4 AlternativeEvent ............................................. 1

1.1.5 AnnotateOutputEvent .......................................... 2

1.1.6 ApplyContext .................................................. 2

1.1.7 ApplyOutputEvent ............................................. 3

1.1.8 ArpeggioEvent ................................................. 3

1.1.9 ArticulationEvent ............................................. 3

1.1.10 BarCheck ..................................................... 4

1.1.11 BarEvent ..................................................... 4

1.1.12 BassFigureEvent ............................................. 5

1.1.13 BeamEvent ................................................... 5

1.1.14 BeamForbidEvent ............................................ 5

1.1.15 BendAfterEvent .............................................. 6

1.1.16 BendSpanEvent ............................................... 6

1.1.17 BreakDynamicSpanEvent ..................................... 6

1.1.18 BreathingEvent ............................................... 7

1.1.19 CaesuraEvent ................................................ 7

1.1.20 ClusterNoteEvent ............................................ 7

1.1.21 CodaMarkEvent .............................................. 8

1.1.22 CompletizeExtenderEvent ................................... 8

1.1.23 ContextChange ............................................... 8

1.1.24 ContextSpeccedMusic ........................................ 9

1.1.25 CrescendoEvent .............................................. 9

1.1.26 DalSegnoEvent ............................................... 10

1.1.27 DecrescendoEvent .......................................... 10

1.1.28 DoublePercentEvent ....................................... 10

1.1.29 DurationLineEvent ......................................... 11

1.1.30 EpisemaEvent ............................................... 11

1.1.31 Event ......................................................... 11

1.1.32 EventChord .................................................. 12

1.1.33 ExtenderEvent .............................................. 12

1.1.34 FineEvent ................................................... 13

1.1.35 FingerGlideEvent .......................................... 13

1.1.36 FingeringEvent ............................................. 13

1.1.37 FootnoteEvent ............................................... 14

1.1.38 GlissandoEvent .............................................. 14

1.1.39 GraceMusic .................................................. 14

1.1.40 HarmonicEvent ............................................. 15

1.1.41 HyphenEvent ............................................... 15

1.1.42 KeyChangeEvent ............................................. 15

1.1.43 LabelEvent .................................................. 16

1.1.44 LaissezVibreEvent ......................................... 16

1.1.45 LigatureEvent .............................................. 16

1.1.46 LineBreakEvent ............................................. 17
1.1.47 LyricCombineMusic .................................................. 17
1.1.48 LyricEvent ............................................................. 18
1.1.49 MeasureCounterEvent ............................................... 18
1.1.50 MeasureSpannerEvent .............................................. 18
1.1.51 MultiMeasureArticulationEvent .................................. 19
1.1.52 MultiMeasureRestEvent ............................................ 19
1.1.53 MultiMeasureRestMusic ............................................. 19
1.1.54 MultiMeasureTextEvent ............................................. 20
1.1.55 Music ................................................................. 20
1.1.56 NoteEvent ............................................................. 20
1.1.57 NoteGroupingEvent .................................................. 21
1.1.58 OttawaEvent .......................................................... 21
1.1.59 OverrideProperty ..................................................... 22
1.1.60 PageBreakEvent ....................................................... 22
1.1.61 PageTurnEvent ......................................................... 22
1.1.62 PartCombineMusic .................................................... 23
1.1.63 PartialSet .............................................................. 23
1.1.64 PercentEvent .......................................................... 23
1.1.65 PercentRepeatedMusic .............................................. 24
1.1.66 PesOrFlexaEvent ....................................................... 24
1.1.67 PhrasingSlurEvent ..................................................... 25
1.1.68 PostEvents ............................................................... 25
1.1.69 PropertySet ............................................................. 25
1.1.70 PropertyUnset .......................................................... 26
1.1.71 QuoteMusic .............................................................. 26
1.1.72 RehearsalMarkEvent .................................................. 27
1.1.73 RelativeOctaveCheck ................................................ 27
1.1.74 RelativeOctaveMusic ................................................ 27
1.1.75 RepeatSlashEvent .................................................... 28
1.1.76 RepeatTieEvent ........................................................ 28
1.1.77 RestEvent ............................................................... 28
1.1.78 RevertProperty ......................................................... 29
1.1.79 ScriptEvent ............................................................. 29
1.1.80 SectionEvent ........................................................... 30
1.1.81 SectionLabelEvent .................................................... 30
1.1.82 SegnoMarkEvent ....................................................... 30
1.1.83 SegnoRepeatedMusic ................................................ 31
1.1.84 SequentialAlternativeMusic ...................................... 31
1.1.85 SequentialMusic ....................................................... 32
1.1.86 SimultaneousMusic ................................................... 32
1.1.87 SkipEvent ............................................................... 33
1.1.88 SkipMusic ................................................................. 33
1.1.89 SkippedMusic ............................................................ 34
1.1.90 SlurEvent ............................................................... 34
1.1.91 SoloOneEvent .......................................................... 35
1.1.92 SoloTwoEvent .......................................................... 35
1.1.93 SostenutoEvent ........................................................ 35
1.1.94 SpacingSectionEvent ................................................ 36
1.1.95 SpanEvent ............................................................... 36
1.1.96 StaffHighlightEvent .................................................. 36
1.1.97 StaffSpanEvent ........................................................ 37
1.1.98 StringNumberEvent ................................................... 37
1.1.99 StrokeFingerEvent ..................................................... 37
1.1.100 SustainEvent .......................................................... 38
1.1.101 TempoChangeEvent ........................................ 38
1.1.102 TextScriptEvent ........................................... 38
1.1.103 TextSpanEvent ............................................. 39
1.1.104 TieEvent .................................................. 39
1.1.105 TimeScaledMusic .......................................... 39
1.1.106 TimeSignatureEvent ....................................... 40
1.1.107 TimeSignatureMusic ....................................... 40
1.1.108 TransposedMusic .......................................... 41
1.1.109 TremoloEvent ............................................. 41
1.1.110 TremoloRepeatedMusic ................................... 41
1.1.111 TremoloSpanEvent ......................................... 42
1.1.112 TrillSpanEvent ........................................... 42
1.1.113 TupleSpanEvent ........................................... 43
1.1.114 UnaCordaEvent ........................................... 43
1.1.115 UnfoldedRepeatedMusic ................................... 43
1.1.116 UnfoldedSpeccedMusic .................................... 44
1.1.117 UnisonoEvent ............................................. 44
1.1.118 UnrelativableMusic ........................................ 45
1.1.119 VoiceSeparator ............................................ 45
1.1.120 VoltaRepeatEndEvent ..................................... 46
1.1.121 VoltaRepeatStartEvent ................................... 46
1.1.122 VoltaRepeatedMusic ....................................... 46
1.1.123 VoltaSpanEvent ........................................... 47
1.1.124 VoltaSpeccedMusic ........................................ 47
1.1.125 VowelTransitionEvent ................................... 48

1.2 Music classes ..................................................... 48
1.2.1 absolute-dynamic-event ................................... 48
1.2.2 ad-hoc-jump-event .......................................... 48
1.2.3 ad-hoc-mark-event ......................................... 48
1.2.4 alternative-event .......................................... 48
1.2.5 annotate-output-event ..................................... 48
1.2.6 apply-output-event ......................................... 48
1.2.7 arpeggio-event ............................................. 49
1.2.8 articulation-event ......................................... 49
1.2.9 bar-event .................................................. 49
1.2.10 bass-figure-event ......................................... 49
1.2.11 beam-event ................................................ 49
1.2.12 beam-forbid-event ....................................... 49
1.2.13 bend-after-event .......................................... 49
1.2.14 bend-span-event .......................................... 49
1.2.15 break-dynamic-span-event ................................. 49
1.2.16 break-event .............................................. 49
1.2.17 break-span-event ......................................... 50
1.2.18 breathing-event .......................................... 50
1.2.19 caesura-event ............................................. 50
1.2.20 cluster-note-event ....................................... 50
1.2.21 coda-mark-event .......................................... 50
1.2.22 completize-extender-event ................................. 50
1.2.23 crescendo-event .......................................... 50
1.2.24 dal-segno-event .......................................... 50
1.2.25 decrescendo-event ....................................... 50
1.2.26 double-percent-event ..................................... 50
1.2.27 duration-line-event ...................................... 51
1.2.28 dynamic-event ............................................ 51
<table>
<thead>
<tr>
<th>Event Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>episema-event</td>
<td>51</td>
</tr>
<tr>
<td>extender-event</td>
<td>51</td>
</tr>
<tr>
<td>fine-event</td>
<td>51</td>
</tr>
<tr>
<td>finger-glide-event</td>
<td>51</td>
</tr>
<tr>
<td>fingering-event</td>
<td>51</td>
</tr>
<tr>
<td>footnote-event</td>
<td>51</td>
</tr>
<tr>
<td>general-rest-event</td>
<td>51</td>
</tr>
<tr>
<td>glissando-event</td>
<td>51</td>
</tr>
<tr>
<td>harmonic-event</td>
<td>51</td>
</tr>
<tr>
<td>hyphen-event</td>
<td>52</td>
</tr>
<tr>
<td>key-change-event</td>
<td>52</td>
</tr>
<tr>
<td>label-event</td>
<td>52</td>
</tr>
<tr>
<td>laissez-vibrer-event</td>
<td>52</td>
</tr>
<tr>
<td>layout-instruction-event</td>
<td>52</td>
</tr>
<tr>
<td>ligature-event</td>
<td>52</td>
</tr>
<tr>
<td>line-break-event</td>
<td>52</td>
</tr>
<tr>
<td>lyric-event</td>
<td>52</td>
</tr>
<tr>
<td>mark-event</td>
<td>52</td>
</tr>
<tr>
<td>measure-counter-event</td>
<td>52</td>
</tr>
<tr>
<td>measure-spanner-event</td>
<td>53</td>
</tr>
<tr>
<td>melodic-event</td>
<td>53</td>
</tr>
<tr>
<td>multi-measure-articulation-event</td>
<td>53</td>
</tr>
<tr>
<td>multi-measure-rest-event</td>
<td>53</td>
</tr>
<tr>
<td>multi-measure-text-event</td>
<td>53</td>
</tr>
<tr>
<td>music-event</td>
<td>53</td>
</tr>
<tr>
<td>note-event</td>
<td>54</td>
</tr>
<tr>
<td>note-grouping-event</td>
<td>54</td>
</tr>
<tr>
<td>ottava-event</td>
<td>54</td>
</tr>
<tr>
<td>page-break-event</td>
<td>54</td>
</tr>
<tr>
<td>page-turn-event</td>
<td>54</td>
</tr>
<tr>
<td>part-combine-event</td>
<td>54</td>
</tr>
<tr>
<td>pedal-event</td>
<td>54</td>
</tr>
<tr>
<td>percent-event</td>
<td>54</td>
</tr>
<tr>
<td>pes-or-flexa-event</td>
<td>55</td>
</tr>
<tr>
<td>phrasing-slur-event</td>
<td>55</td>
</tr>
<tr>
<td>rehearsal-mark-event</td>
<td>55</td>
</tr>
<tr>
<td>repeat-slash-event</td>
<td>55</td>
</tr>
<tr>
<td>repeat-tie-event</td>
<td>55</td>
</tr>
<tr>
<td>rest-event</td>
<td>55</td>
</tr>
<tr>
<td>rhythmic-event</td>
<td>55</td>
</tr>
<tr>
<td>script-event</td>
<td>55</td>
</tr>
<tr>
<td>section-event</td>
<td>55</td>
</tr>
<tr>
<td>section-label-event</td>
<td>55</td>
</tr>
<tr>
<td>segno-mark-event</td>
<td>56</td>
</tr>
<tr>
<td>skip-event</td>
<td>56</td>
</tr>
<tr>
<td>slur-event</td>
<td>56</td>
</tr>
<tr>
<td>solo-one-event</td>
<td>56</td>
</tr>
<tr>
<td>solo-two-event</td>
<td>56</td>
</tr>
<tr>
<td>sostenuto-event</td>
<td>56</td>
</tr>
<tr>
<td>spacing-section-event</td>
<td>56</td>
</tr>
<tr>
<td>span-dynamic-event</td>
<td>56</td>
</tr>
<tr>
<td>span-event</td>
<td>56</td>
</tr>
<tr>
<td>staff-highlight-event</td>
<td>56</td>
</tr>
<tr>
<td>staff-span-event</td>
<td>57</td>
</tr>
</tbody>
</table>
2 Translation ................................................................. 66
  2.1 Contexts ............................................................ 66
    2.1.1 ChoirStaff ................................................ 66
    2.1.2 ChordGrid ................................................ 68
    2.1.3 ChordGridScore .......................................... 73
    2.1.4 ChordNames .............................................. 95
    2.1.5 CueVoice .................................................. 97
    2.1.6 Devnull .................................................. 108
    2.1.7 DrumStaff ................................................. 108
    2.1.8 DrumVoice ................................................ 117
    2.1.9 Dynamics ................................................. 126
    2.1.10 FiguredBass ............................................ 132
    2.1.11 FretBoards ............................................. 133
    2.1.12 Global .................................................. 135
    2.1.13 GrandStaff ............................................. 135
    2.1.14 GregorianTranscriptionLyrics ......................... 137
    2.1.15 GregorianTranscriptionStaff ......................... 140
    2.1.16 GregorianTranscriptionVoice ......................... 153
    2.1.17 InternalGregorianStaff ................................. 163
    2.1.18 KievanStaff ............................................ 176
    2.1.19 KievanVoice ............................................ 189
    2.1.20 Lyrics .................................................. 199
    2.1.21 MensuralStaff .......................................... 202
    2.1.22 MensuralVoice ......................................... 216
    2.1.23 NoteNames ............................................. 226
    2.1.24 NullVoice ............................................... 228
    2.1.25 OneStaff ................................................. 230
    2.1.26 PetrucciStaff .......................................... 231
    2.1.27 PetrucciVoice .......................................... 244
    2.1.28 PianoStaff ............................................. 255
    2.1.29 RhythmicStaff ......................................... 257
    2.1.30 Score .................................................... 263
    2.1.31 Staff .................................................... 287
  2.2 Menus ............................................................. 287
  2.3 Other representations ........................................ 287
  2.4 Implementations ................................................ 291
  2.5 Bibliography ................................................... 308
### 2.2 Engravers and Performers

<table>
<thead>
<tr>
<th>Engraver/Performer</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accidental_engraver</td>
<td>400</td>
</tr>
<tr>
<td>Alteration_glyph_engraver</td>
<td>401</td>
</tr>
<tr>
<td>Ambitus_engraver</td>
<td>402</td>
</tr>
<tr>
<td>Arpeggio_engraver</td>
<td>402</td>
</tr>
<tr>
<td>Auto_beam_engraver</td>
<td>402</td>
</tr>
<tr>
<td>Axis_group_engraver</td>
<td>403</td>
</tr>
<tr>
<td>Balloon_engraver</td>
<td>404</td>
</tr>
<tr>
<td>Bar_engraver</td>
<td>404</td>
</tr>
<tr>
<td>Bar_number_engraver</td>
<td>406</td>
</tr>
<tr>
<td>Beam_collision_engraver</td>
<td>407</td>
</tr>
<tr>
<td>Beam_engraver</td>
<td>407</td>
</tr>
<tr>
<td>Beam_performer</td>
<td>408</td>
</tr>
<tr>
<td>Beat_engraver</td>
<td>408</td>
</tr>
<tr>
<td>Beat_performer</td>
<td>409</td>
</tr>
<tr>
<td>Bend_engraver</td>
<td>409</td>
</tr>
<tr>
<td>Bend_spanner_engraver</td>
<td>410</td>
</tr>
<tr>
<td>Break_align_engraver</td>
<td>410</td>
</tr>
<tr>
<td>Breathing_sign_engraver</td>
<td>410</td>
</tr>
<tr>
<td>Caesura_engraver</td>
<td>410</td>
</tr>
<tr>
<td>Centered_bar_number_align_engraver</td>
<td>411</td>
</tr>
<tr>
<td>Chord_name_engraver</td>
<td>412</td>
</tr>
<tr>
<td>Chord_square_engraver</td>
<td>412</td>
</tr>
<tr>
<td>Chord_tremolo_engraver</td>
<td>412</td>
</tr>
<tr>
<td>Clef_engraver</td>
<td>413</td>
</tr>
<tr>
<td>Cluster_spanner_engraver</td>
<td>413</td>
</tr>
<tr>
<td>Collision_engraver</td>
<td>413</td>
</tr>
<tr>
<td>Completion_heads_engraver</td>
<td>414</td>
</tr>
<tr>
<td>Completion_rest_engraver</td>
<td>414</td>
</tr>
<tr>
<td>Concurrent_hairpin_engraver</td>
<td>415</td>
</tr>
<tr>
<td>Control_track_performer</td>
<td>415</td>
</tr>
<tr>
<td>Cue_clef_engraver</td>
<td>415</td>
</tr>
<tr>
<td>Current_chord_text_engraver</td>
<td>416</td>
</tr>
<tr>
<td>Custos_engraver</td>
<td>417</td>
</tr>
<tr>
<td>Divisi_engraver</td>
<td>417</td>
</tr>
<tr>
<td>Dot_column_engraver</td>
<td>418</td>
</tr>
<tr>
<td>Dots_engraver</td>
<td>418</td>
</tr>
<tr>
<td>Double_percent_repeat_engraver</td>
<td>418</td>
</tr>
<tr>
<td>Drum_note_performer</td>
<td>418</td>
</tr>
<tr>
<td>Drum_notes_engraver</td>
<td>419</td>
</tr>
<tr>
<td>Duration_line_engraver</td>
<td>419</td>
</tr>
<tr>
<td>Dynamic_align_engraver</td>
<td>419</td>
</tr>
<tr>
<td>Dynamic_engraver</td>
<td>420</td>
</tr>
<tr>
<td>Dynamic_performer</td>
<td>420</td>
</tr>
</tbody>
</table>
3 Backend ............................................. 475
  3.1 All layout objects .................................. 475
    3.1.1 Accidental ...................................... 475
    3.1.2 AccidentalCautionary .............................. 476
    3.1.3 AccidentalPlacement .............................. 477
    3.1.4 AccidentalSuggestion ............................. 478
    3.1.5 Ambitus ........................................ 479
    3.1.6 AmbitusAccidental ............................... 481
    3.1.7 AmbitusLine ..................................... 481
    3.1.8 AmbitusNoteHead ................................ 482
    3.1.9 Arpeggio ........................................ 483
    3.1.10 BalloonText .................................... 484
    3.1.11 BarLine ......................................... 486
    3.1.12 BarNumber ...................................... 489
    3.1.13 BassFigure ...................................... 491
    3.1.14 BassFigureAlignment ......................... 492
    3.1.15 BassFigureAlignmentPositioning .............. 492
    3.1.16 BassFigureBracket ............................. 493
    3.1.17 BassFigureContinuation ....................... 494
    3.1.18 BassFigureLine ................................ 494
    3.1.19 Beam ........................................... 495
    3.1.20 BendAfter ...................................... 497
    3.1.21 BendSpanner .................................... 498
    3.1.22 BreakAlignGroup ................................ 500
    3.1.23 BreakAlignment ................................ 501
    3.1.24 BreathingSign .................................. 502
    3.1.25 CaesuraScript .................................. 504
    3.1.26 CenteredBarNumber ............................. 506
    3.1.27 CenteredBarNumberLineSpanner ............... 507
    3.1.28 ChordName ...................................... 508
    3.1.29 ChordSquare .................................... 509
    3.1.30 Clef ............................................ 510
    3.1.31 ClefModifier ................................... 513
    3.1.32 ClusterSpanner ................................ 514
    3.1.33 ClusterSpannerBeacon ......................... 515
    3.1.34 CodaMark ....................................... 515
    3.1.35 CombineTextScript .............................. 517
    3.1.36 ControlPoint ................................... 519
    3.1.37 ControlPolygon ................................ 520
    3.1.38 CueClef ......................................... 521
    3.1.39 CueEndClef ..................................... 524
    3.1.40 Custos .......................................... 527
    3.1.41 Divisio ......................................... 528
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.42 DotColumn</td>
<td>531</td>
</tr>
<tr>
<td>3.1.43 Dots</td>
<td>531</td>
</tr>
<tr>
<td>3.1.44 DoublePercentRepeat</td>
<td>532</td>
</tr>
<tr>
<td>3.1.45 DoublePercentRepeatCounter</td>
<td>533</td>
</tr>
<tr>
<td>3.1.46 DoubleRepeatSlash</td>
<td>535</td>
</tr>
<tr>
<td>3.1.47 DurationLine</td>
<td>536</td>
</tr>
<tr>
<td>3.1.48 DynamicLineSpanner</td>
<td>538</td>
</tr>
<tr>
<td>3.1.49 DynamicText</td>
<td>539</td>
</tr>
<tr>
<td>3.1.50 DynamicTextSpanner</td>
<td>541</td>
</tr>
<tr>
<td>3.1.51 Episema</td>
<td>542</td>
</tr>
<tr>
<td>3.1.52 FingerGlideSpanner</td>
<td>543</td>
</tr>
<tr>
<td>3.1.53 Fingering</td>
<td>545</td>
</tr>
<tr>
<td>3.1.54 FingeringColumn</td>
<td>547</td>
</tr>
<tr>
<td>3.1.55 Flag</td>
<td>547</td>
</tr>
<tr>
<td>3.1.56 Footnote</td>
<td>548</td>
</tr>
<tr>
<td>3.1.57 FretBoard</td>
<td>550</td>
</tr>
<tr>
<td>3.1.58 Glissando</td>
<td>552</td>
</tr>
<tr>
<td>3.1.59 GraceSpacing</td>
<td>553</td>
</tr>
<tr>
<td>3.1.60 GridChordName</td>
<td>553</td>
</tr>
<tr>
<td>3.1.61 GridLine</td>
<td>554</td>
</tr>
<tr>
<td>3.1.62 GridPoint</td>
<td>555</td>
</tr>
<tr>
<td>3.1.63 Hairpin</td>
<td>555</td>
</tr>
<tr>
<td>3.1.64 HorizontalBracket</td>
<td>557</td>
</tr>
<tr>
<td>3.1.65 HorizontalBracketText</td>
<td>558</td>
</tr>
<tr>
<td>3.1.66 InstrumentName</td>
<td>559</td>
</tr>
<tr>
<td>3.1.67 InstrumentSwitch</td>
<td>560</td>
</tr>
<tr>
<td>3.1.68 JumpScript</td>
<td>561</td>
</tr>
<tr>
<td>3.1.69 KeyCancellation</td>
<td>563</td>
</tr>
<tr>
<td>3.1.70 KeySignature</td>
<td>565</td>
</tr>
<tr>
<td>3.1.71 KievanLigature</td>
<td>568</td>
</tr>
<tr>
<td>3.1.72 LaissezVibrerTie</td>
<td>569</td>
</tr>
<tr>
<td>3.1.73 LaissezVibrerTieColumn</td>
<td>570</td>
</tr>
<tr>
<td>3.1.74 LedgerLineSpanner</td>
<td>570</td>
</tr>
<tr>
<td>3.1.75 LeftEdge</td>
<td>571</td>
</tr>
<tr>
<td>3.1.76 LigatureBracket</td>
<td>573</td>
</tr>
<tr>
<td>3.1.77 LyricExtender</td>
<td>574</td>
</tr>
<tr>
<td>3.1.78 LyricHyphen</td>
<td>575</td>
</tr>
<tr>
<td>3.1.79 LyricRepeatCount</td>
<td>576</td>
</tr>
<tr>
<td>3.1.80 LyricSpace</td>
<td>578</td>
</tr>
<tr>
<td>3.1.81 LyricText</td>
<td>579</td>
</tr>
<tr>
<td>3.1.82 MeasureCounter</td>
<td>580</td>
</tr>
<tr>
<td>3.1.83 MeasureGrouping</td>
<td>582</td>
</tr>
<tr>
<td>3.1.84 MeasureSpanner</td>
<td>583</td>
</tr>
<tr>
<td>3.1.85 MelodyItem</td>
<td>585</td>
</tr>
<tr>
<td>3.1.86 MensuralLigature</td>
<td>585</td>
</tr>
<tr>
<td>3.1.87 MetronomeMark</td>
<td>585</td>
</tr>
<tr>
<td>3.1.88 MultiMeasureRest</td>
<td>587</td>
</tr>
<tr>
<td>3.1.89 MultiMeasureRestNumber</td>
<td>589</td>
</tr>
<tr>
<td>3.1.90 MultiMeasureRestScript</td>
<td>590</td>
</tr>
<tr>
<td>3.1.91 MultiMeasureRestText</td>
<td>592</td>
</tr>
<tr>
<td>3.1.92 NonMusicalPaperColumn</td>
<td>593</td>
</tr>
<tr>
<td>3.1.93 NoteCollision</td>
<td>595</td>
</tr>
<tr>
<td>3.1.94 NoteColumn</td>
<td>595</td>
</tr>
<tr>
<td>3.1.95 NoteHead</td>
<td>596</td>
</tr>
</tbody>
</table>
3.1.96 NoteName ................................................................. 597
3.1.97 NoteSpacing .............................................................. 598
3.1.98 OttavaBracket ........................................................... 599
3.1.99 PaperColumn ............................................................. 600
3.1.100 Parentheses ............................................................ 601
3.1.101 PercentRepeat .......................................................... 602
3.1.102 PercentRepeatCounter ................................................ 603
3.1.103 PhrasingSlur ............................................................ 604
3.1.104 PianoPedalBracket .................................................... 606
3.1.105 RehearsalMark ......................................................... 607
3.1.106 RepeatSlash ............................................................ 609
3.1.107 RepeatTie ............................................................... 610
3.1.108 RepeatTieColumn ....................................................... 611
3.1.109 Rest ...................................................................... 611
3.1.110 RestCollision ........................................................... 612
3.1.111 Script ................................................................. 613
3.1.112 ScriptColumn ........................................................... 614
3.1.113 ScriptRow .............................................................. 614
3.1.114 SectionLabel ............................................................ 615
3.1.115 SegnoMark .............................................................. 616
3.1.116 SignumRepetitionis .................................................... 618
3.1.117 Slur ................................................................. 621
3.1.118 SostenutoPedal ........................................................ 623
3.1.119 SostenutoPedallineSpanner ........................................ 624
3.1.120 SpacingSpanner ........................................................ 626
3.1.121 SpanBar ............................................................... 626
3.1.122 SpanBarStub ............................................................ 628
3.1.123 StaffEllipsis ............................................................. 628
3.1.124 StaffGrouper ............................................................ 630
3.1.125 StaffHighlight .......................................................... 631
3.1.126 StaffSpacing ............................................................ 632
3.1.127 StaffSymbol ............................................................ 632
3.1.128 StanzaNumber .......................................................... 633
3.1.129 Stem ................................................................. 634
3.1.130 StemStub ............................................................... 636
3.1.131 StemTremolo ............................................................ 637
3.1.132 StringNumber ........................................................... 638
3.1.133 StrokeFinger ............................................................. 640
3.1.134 SustainPedal ............................................................ 641
3.1.135 SustainPedallineSpanner ............................................ 642
3.1.136 System ............................................................... 643
3.1.137 SystemStartBar ........................................................ 644
3.1.138 SystemStartBrace ...................................................... 645
3.1.139 SystemStartBracket .................................................. 646
3.1.140 SystemStartSquare .................................................... 647
3.1.141 TabNoteHead ............................................................ 648
3.1.142 TextScript ............................................................. 650
3.1.143 TextSpanner ............................................................. 652
3.1.144 Tie ................................................................. 653
3.1.145 TieColumn ............................................................. 655
3.1.146 TimeSignature .......................................................... 655
3.1.147 TrillPitchAccidental .................................................. 657
3.1.148 TrillPitchGroup ........................................................ 658
3.1.149 TrillPitchHead ........................................................ 659
3.2 Graphical Object Interfaces

3.2.1 accidental-interface ............................................. 675
3.2.2 accidental-placement-interface ................................. 675
3.2.3 accidental-suggestion-interface ................................. 676
3.2.4 accidental-switch-interface .................................... 676
3.2.5 align-interface .................................................. 676
3.2.6 ambitus-interface ................................................ 677
3.2.7 arpeggio-interface .............................................. 677
3.2.8 axis-group-interface ........................................... 678
3.2.9 balloon-interface ................................................ 680
3.2.10 bar-line-interface ............................................... 681
3.2.11 bar-number-interface .......................................... 682
3.2.12 bass-figure-alignment-interface ............................... 682
3.2.13 bass-figure-interface .......................................... 682
3.2.14 beam-interface ................................................. 682
3.2.15 bend-after-interface .......................................... 685
3.2.16 bend-interface .................................................. 685
3.2.17 bezier-curve-interface ....................................... 687
3.2.18 break-alignable-interface .................................... 687
3.2.19 break-aligned-interface ....................................... 687
3.2.20 break-alignment-interface .................................... 688
3.2.21 breathing-sign-interface ...................................... 689
3.2.22 caesura-script-interface ...................................... 690
3.2.23 centered-bar-number-interface ................................. 690
3.2.24 centered-bar-number-line-spanner-interface .................... 690
3.2.25 centered-spanner-interface .................................... 690
3.2.26 chord-name-interface .......................................... 691
3.2.27 chord-square-interface ....................................... 691
3.2.28 clef-interface .................................................. 691
3.2.29 clef-modifier-interface ....................................... 692
3.2.30 cluster-beacon-interface ..................................... 692
3.2.31 cluster-interface ............................................... 692
3.2.32 coda-mark-interface .......................................... 692
3.2.33 control-point-interface ....................................... 693
3.2.34 control-polygon-interface .................................... 693
3.2.35 custos-interface ................................................. 693
3.2.36 dot-column-interface ......................................... 693
3.2.37 dots-interface .................................................. 694
3.2.38 duration-line-interface ....................................... 694
3.2.39 dynamic-interface ............................................. 695
3.2.40 dynamic-line-spanner-interface ............................... 695
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.41</td>
<td>dynamic-text-interface</td>
</tr>
<tr>
<td>3.2.42</td>
<td>dynamic-text-spanner-interface</td>
</tr>
<tr>
<td>3.2.43</td>
<td>enclosing-bracket-interface</td>
</tr>
<tr>
<td>3.2.44</td>
<td>episema-interface</td>
</tr>
<tr>
<td>3.2.45</td>
<td>figured-bass-continuation-interface</td>
</tr>
<tr>
<td>3.2.46</td>
<td>finger-glide-interface</td>
</tr>
<tr>
<td>3.2.47</td>
<td>finger-interface</td>
</tr>
<tr>
<td>3.2.48</td>
<td>fingering-column-interface</td>
</tr>
<tr>
<td>3.2.49</td>
<td>flag-interface</td>
</tr>
<tr>
<td>3.2.50</td>
<td>font-interface</td>
</tr>
<tr>
<td>3.2.51</td>
<td>footnote-interface</td>
</tr>
<tr>
<td>3.2.52</td>
<td>fret-diagram-interface</td>
</tr>
<tr>
<td>3.2.53</td>
<td>glissando-interface</td>
</tr>
<tr>
<td>3.2.54</td>
<td>grace-spacing-interface</td>
</tr>
<tr>
<td>3.2.55</td>
<td>gregorian-ligature-interface</td>
</tr>
<tr>
<td>3.2.56</td>
<td>grid-chord-name-interface</td>
</tr>
<tr>
<td>3.2.57</td>
<td>grid-line-interface</td>
</tr>
<tr>
<td>3.2.58</td>
<td>grid-point-interface</td>
</tr>
<tr>
<td>3.2.59</td>
<td>grob-interface</td>
</tr>
<tr>
<td>3.2.60</td>
<td>hairpin-interface</td>
</tr>
<tr>
<td>3.2.61</td>
<td>hara-kiri-group-spanner-interface</td>
</tr>
<tr>
<td>3.2.62</td>
<td>horizontal-bracket-interface</td>
</tr>
<tr>
<td>3.2.63</td>
<td>horizontal-bracket-text-interface</td>
</tr>
<tr>
<td>3.2.64</td>
<td>horizontal-line-spanner-interface</td>
</tr>
<tr>
<td>3.2.65</td>
<td>inline-accidental-interface</td>
</tr>
<tr>
<td>3.2.66</td>
<td>instrument-specific-markup-interface</td>
</tr>
<tr>
<td>3.2.67</td>
<td>item-interface</td>
</tr>
<tr>
<td>3.2.68</td>
<td>jump-script-interface</td>
</tr>
<tr>
<td>3.2.69</td>
<td>key-cancellation-interface</td>
</tr>
<tr>
<td>3.2.70</td>
<td>key-signature-interface</td>
</tr>
<tr>
<td>3.2.71</td>
<td>kievan-ligature-interface</td>
</tr>
<tr>
<td>3.2.72</td>
<td>ledger-line-spanner-interface</td>
</tr>
<tr>
<td>3.2.73</td>
<td>ledgered-interface</td>
</tr>
<tr>
<td>3.2.74</td>
<td>ligature-bracket-interface</td>
</tr>
<tr>
<td>3.2.75</td>
<td>ligature-head-interface</td>
</tr>
<tr>
<td>3.2.76</td>
<td>ligature-interface</td>
</tr>
<tr>
<td>3.2.77</td>
<td>line-interface</td>
</tr>
<tr>
<td>3.2.78</td>
<td>line-spanner-interface</td>
</tr>
<tr>
<td>3.2.79</td>
<td>lyric-extender-interface</td>
</tr>
<tr>
<td>3.2.80</td>
<td>lyric-hyphen-interface</td>
</tr>
<tr>
<td>3.2.81</td>
<td>lyric-interface</td>
</tr>
<tr>
<td>3.2.82</td>
<td>lyric-repeat-count-interface</td>
</tr>
<tr>
<td>3.2.83</td>
<td>lyric-space-interface</td>
</tr>
<tr>
<td>3.2.84</td>
<td>lyric-syllable-interface</td>
</tr>
<tr>
<td>3.2.85</td>
<td>mark-interface</td>
</tr>
<tr>
<td>3.2.86</td>
<td>measure-counter-interface</td>
</tr>
<tr>
<td>3.2.87</td>
<td>measure-grouping-interface</td>
</tr>
<tr>
<td>3.2.88</td>
<td>measure-spanner-interface</td>
</tr>
<tr>
<td>3.2.89</td>
<td>melody-spanner-interface</td>
</tr>
<tr>
<td>3.2.90</td>
<td>mensural-ligature-interface</td>
</tr>
<tr>
<td>3.2.91</td>
<td>metronome-mark-interface</td>
</tr>
<tr>
<td>3.2.92</td>
<td>multi-measure-interface</td>
</tr>
<tr>
<td>3.2.93</td>
<td>multi-measure-rest-interface</td>
</tr>
<tr>
<td>3.2.94</td>
<td>multi-measure-rest-number-interface</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.2.149</td>
<td>754</td>
</tr>
<tr>
<td>system-start-text-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.150</td>
<td>754</td>
</tr>
<tr>
<td>tab-note-head-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.151</td>
<td>755</td>
</tr>
<tr>
<td>text-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.152</td>
<td>756</td>
</tr>
<tr>
<td>text-script-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.153</td>
<td>756</td>
</tr>
<tr>
<td>tie-column-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.154</td>
<td>757</td>
</tr>
<tr>
<td>tie-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.155</td>
<td>759</td>
</tr>
<tr>
<td>time-signature-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.156</td>
<td>760</td>
</tr>
<tr>
<td>trill-pitch-accidental-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.157</td>
<td>760</td>
</tr>
<tr>
<td>trill-spanner-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.158</td>
<td>760</td>
</tr>
<tr>
<td>tuplet-bracket-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.159</td>
<td>761</td>
</tr>
<tr>
<td>tuplet-number-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.160</td>
<td>762</td>
</tr>
<tr>
<td>unbreakable-spanner-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.161</td>
<td>762</td>
</tr>
<tr>
<td>vaticana-ligature-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.162</td>
<td>763</td>
</tr>
<tr>
<td>volta-bracket-interface</td>
<td></td>
</tr>
<tr>
<td>3.2.163</td>
<td>763</td>
</tr>
<tr>
<td>volta-interface</td>
<td></td>
</tr>
<tr>
<td>3.3 User backend properties</td>
<td>764</td>
</tr>
<tr>
<td>3.4 Internal backend properties</td>
<td>786</td>
</tr>
</tbody>
</table>

4 **Scheme functions** .................................................. 794

Appendix A  Indices ..................................................... 849

A.1 Concept index ....................................................... 849
A.2 Function index ....................................................... 849
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: absolute-dynamic-event (page 48), dynamic-event (page 51),
music-event (page 53), and StreamEvent (page 57).

Accepted by: Dynamic_engraver (page 420), and Dynamic_performer (page 420).

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 AdHocJumpEvent
Insert a JumpScript.

Syntax: \jump markup

Example: \jump "Gavotte I D.C."

Event classes: ad-hoc-jump-event (page 48), music-event (page 53), and StreamEvent (page 57).

Accepted by: Bar_engraver (page 404), and Jump_engraver (page 428).

Properties:
- name (symbol): 'AdHocJumpEvent
  Name of this music object.
- types (list):
  '(ad-hoc-jump-event event)
  The types of this music object; determines by what engraver this music expression is processed.

1.1.3 AdHocMarkEvent
Insert markup as a rehearsal mark without advancing the rehearsal mark sequence.

Syntax: \mark markup

Example: \mark "A"

Event classes: ad-hoc-mark-event (page 48), mark-event (page 52), music-event (page 53), and StreamEvent (page 57).
Chapter 1: Music definitions

Accepted by: Mark_trackingTranslator (page 433).

Properties:

- name (symbol): 'AdHocMarkEvent
  Name of this music object.

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

1.1.4 AlternativeEvent

Create an alternative event.

- Event classes: alternative-event (page 48), music-event (page 53), and StreamEvent (page 57).

- Accepted by: TimingTranslator (page 454).

- 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.

1.1.5 AnnotateOutputEvent

Print an annotation of an output element.

- Event classes: annotate-output-event (page 48), music-event (page 53), and StreamEvent (page 57).

- Accepted by: BalloonEngraver (page 404).

- 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.

1.1.6 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.

1.1.7 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: apply-output-event (page 48), layout-instruction-event (page 52),
                    music-event (page 53), and StreamEvent (page 57).
   Accepted by: Output_property_engraver (page 439).
   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.

1.1.8 ArpeggioEvent
Make an arpeggio on this note.
   Syntax: note-\arpeggio
   Event classes: arpeggio-event (page 49), music-event (page 53), and StreamEvent (page 57).
   Accepted by: Arpeggio_engraver (page 402).
   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.

1.1.9 ArticulationEvent
Add an articulation marking to a note.
   Syntax: notexy, 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: articulation-event (page 49), music-event (page 53), script-event (page 55), and StreamEvent (page 57).

Accepted by: Beat engraver (page 408), Beat performer (page 409), Drum note performer (page 418), Note performer (page 439), and Script engraver (page 445).

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.

1.1.10 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.

1.1.11 BarEvent
Force a bar line.

Syntax: \bar type
Example: \bar "!"

Event classes: bar-event (page 49), music-event (page 53), and StreamEvent (page 57).

Accepted by: Timing translator (page 454).

Properties:

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

types (list):
  '(bar-event event)
  The types of this music object; determines by what engraver this music expression is processed.
1.1.12 **BassFigureEvent**

Print a bass-figure text.

- **Event classes:** bass-figure-event (page 49), music-event (page 53), rhythmic-event (page 55), and StreamEvent (page 57).
- **Accepted by:** Figured_bass_engraver (page 421).
- **Properties:**
  - name (symbol): `BassFigureEvent`
  - types (list): `(event rhythmic-event bass-figure-event)`

1.1.13 **BeamEvent**

Start or stop a beam.

- **Syntax for manual control:** c8- [ c c-] c8
- **Event classes:** beam-event (page 49), music-event (page 53), span-event (page 56), and StreamEvent (page 57).
- **Accepted by:** Beam_engraver (page 407), Beam_performer (page 408), and Grace_beam_engraver (page 425).
- **Properties:**
  - name (symbol): `BeamEvent`
  - types (list): `(post-event event beam-event span-event)`

1.1.14 **BeamForbidEvent**

Specify that a note may not auto-beamed.

- **Event classes:** beam-forbid-event (page 49), music-event (page 53), and StreamEvent (page 57).
- **Accepted by:** Auto_beam_engraver (page 402), and Grace_auto_beam_engraver (page 424).
- **Properties:**
  - name (symbol): `BeamForbidEvent`
  - types (list): `(post-event event beam-forbid-event)`
1.1.15 **BendAfterEvent**

A drop/fall/doit jazz articulation.

- Event classes: bend-after-event (page 49), music-event (page 53), and StreamEvent (page 57).
- Accepted by: Bend_engraver (page 409).
- 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.

1.1.16 **BendSpanEvent**

Used to signal where a bend spanner starts and stops.

- Event classes: bend-span-event (page 49), music-event (page 53), span-event (page 56), and StreamEvent (page 57).
- Accepted by: Bend_spanner_engraver (page 410).
- Properties:
  - name (symbol): 'BendSpanEvent
    Name of this music object.
  - types (list):
    - '(bend-span-event post-event span-event event)
      The types of this music object; determines by what engraver this music expression is processed.

1.1.17 **BreakDynamicSpanEvent**

End an alignment spanner for dynamics here.

- Event classes: break-dynamic-span-event (page 49), break-span-event (page 50), music-event (page 53), and StreamEvent (page 57).
- Accepted by: Dynamic_engraver (page 420).
- 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.
1.1.18 **BreathingEvent**

A short span of silence that shortens the previous note.

Syntax: `note`\breathe

Event classes: breathing-event (page 50), music-event (page 53), and StreamEvent (page 57).

Accepted by: Breathing_sign_engraver (page 410), and Note_performer (page 439).

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.

1.1.19 **CaesuraEvent**

A short span of silence that does not shorten the previous note.

Syntax: `note`\caesura

Event classes: caesura-event (page 50), music-event (page 53), and StreamEvent (page 57).

Accepted by: Bar_engraver (page 404), Caesura_engraver (page 410), and Divisio_engraver (page 417).

Properties:

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

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

1.1.20 **ClusterNoteEvent**

A note that is part of a cluster.

Event classes: cluster-note-event (page 50), melodic-event (page 53), music-event (page 53), rhythmic-event (page 55), and StreamEvent (page 57).

Accepted by: Cluster_spanner_engraver (page 413).

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.

1.1.21 CodaMarkEvent
Add a coda mark.
  Event classes: coda-mark-event (page 50), music-event (page 53), and StreamEvent (page 57).
  Accepted by: Bar_engraver (page 404), and Mark_tracking_translator (page 433).
  Properties:
    name (symbol):
      'CodaMarkEvent
      Name of this music object.
    types (list):
      '(coda-mark-event event)
      The types of this music object; determines by what engraver this music expression is processed.

1.1.22 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: completize-extender-event (page 50), music-event (page 53), and StreamEvent (page 57).
  Accepted by: Extender_engraver (page 421).
  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.

1.1.23 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.

1.1.24 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.

1.1.25 CrescendoEvent
Begin or end a crescendo.

  Syntax: note\< ... note\>!
  An alternative syntax is note\cr ... note\endcr.

  Event classes: crescendo-event (page 50), music-event (page 53), span-dynamic-event (page 56), span-event (page 56), and StreamEvent (page 57).

  Accepted by: Dynamic_engraver (page 420), and Dynamic_performer (page 420).

  Properties:

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

  types (list):
  '(post-event
    span-event
    span-dynamic-event
Chapter 1: Music definitions

1.1.26 DalSegnoEvent
Add a D.S. or similar instruction.

Event classes: dal-segno-event (page 50), music-event (page 53), and StreamEvent (page 57).

Accepted by: Bar_engraver (page 404), Jump_engraver (page 428), and Volta_engraver (page 456).

Properties:

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

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

1.1.27 DecrescendoEvent
Begin or end a decrescendo.

Syntax: note> ... note!

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

Event classes: decrescendo-event (page 50), music-event (page 53), span-dynamic-event (page 56), span-event (page 56), and StreamEvent (page 57).

Accepted by: Dynamic_engraver (page 420), and Dynamic_performer (page 420).

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.

1.1.28 DoublePercentEvent
Used internally to signal double percent repeats.

Event classes: double-percent-event (page 50), music-event (page 53), rhythmic-event (page 55), and StreamEvent (page 57).

Accepted by: Double_percent_repeat_engraver (page 418).
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.

1.1.29 DurationLineEvent

Initiate a duration line.

Syntax: note\-

Event classes: duration-line-event (page 51), music-event (page 53), and StreamEvent (page 57).

Accepted by: Duration_line_engraver (page 419).

Properties:

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

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

1.1.30 EpisemaEvent

Begin or end an episema.

Event classes: episema-event (page 51), music-event (page 53), span-event (page 56), and StreamEvent (page 57).

Accepted by: Episema_engraver (page 421).

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.

1.1.31 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.

1.1.32 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.

1.1.33 ExtenderEvent
Extend lyrics.

  Event classes: extender-event (page 51), music-event (page 53), and StreamEvent (page 57).

  Accepted by: Extender_engraver (page 421).

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.
1.1.34 FineEvent

End the performance, not necessarily at the written end of the music.

Event classes: fine-event (page 51), music-event (page 53), and StreamEvent (page 57).
Accepted by: Bar_engraver (page 404), Divisio_engraver (page 417), Jump_engraver (page 428), Timing_translator (page 454), and Volta_engraver (page 456).

Properties:

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

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

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

1.1.35 FingerGlideEvent

Initiate a line connecting two equal fingerings. This line represents a finger gliding on a string.

Syntax: note\glide-finger

Event classes: finger-glide-event (page 51), music-event (page 53), span-event (page 56), and StreamEvent (page 57).
Not accepted by any engraver or performer.

Properties:

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

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

1.1.36 FingeringEvent

Specify what finger to use for this note.

Event classes: fingering-event (page 51), music-event (page 53), and StreamEvent (page 57).
Accepted by: Fingering_engraver (page 422), Fretboard_engraver (page 423), and Tab_note_heads_engraver (page 451).

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.
1.1.37 **FootnoteEvent**

Footnote a grob.

Event classes: footnote-event (page 51), music-event (page 53), and StreamEvent (page 57).

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.

1.1.38 **GlissandoEvent**

Start a glissando on this note.

Event classes: glissando-event (page 51), music-event (page 53), and StreamEvent (page 57).

Accepted by: Glissando_engraver (page 424).

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.

1.1.39 **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 endpoint of this music. This property is unhappily named in that it does not account for any initial grace notes: the full length of the music is length minus the start time. A value of INF-MOMENT indicates indefinite length.

- 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.

1.1.40 HarmonicEvent
Mark a note as harmonic.

  Event classes: harmonic-event (page 51), music-event (page 53), and StreamEvent (page 57).
  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.

1.1.41 HyphenEvent
A hyphen between lyric syllables.

  Event classes: hyphen-event (page 52), music-event (page 53), and StreamEvent (page 57).
  Accepted by: Hyphen_engraver (page 427).

  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.

1.1.42 KeyChangeEvent
Change the key signature.

  Syntax: \key name scale
  
  Event classes: key-change-event (page 52), music-event (page 53), and StreamEvent (page 57).
  Accepted by: Key_engraver (page 429), and Key_performer (page 430).

  Properties:
  
  name (symbol):
    'KeyChangeEvent
    Name of this music object.
to-relative-callback (procedure):
   #<procedure 7fca1bfe19f8 at /build/out/share/lilypond/current/scm/lily/define-music-types.scm:327:33 (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.

1.1.43 LabelEvent
Place a bookmarking label.

   Event classes: label-event (page 52), music-event (page 53), and StreamEvent (page 57).
   Accepted by: Paper_column_engraver (page 440).
   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.

1.1.44 LaissezVibrerEvent
Don’t damp this chord.

   Syntax: note\laissezVibrer
   Event classes: laissez-vibrer-event (page 52), music-event (page 53), and StreamEvent (page 57).
   Accepted by: Laissez_vibrer_engraver (page 430).
   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.

1.1.45 LigatureEvent
Start or end a ligature.

   Event classes: ligature-event (page 52), music-event (page 53), span-event (page 56), and StreamEvent (page 57).
   Accepted by: Kievian_ligature_engraver (page 430), Ligature_bracket_engraver (page 431), Mensural_ligature_engraver (page 435), and Vaticana_ligature_engraver (page 456).
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.

1.1.46 LineBreakEvent
Allow, forbid or force a line break.

Event classes: break-event (page 49), line-break-event (page 52), music-event (page 53), and StreamEvent (page 57).

Accepted by: Page_turn_engraver (page 440), and Paper_column_engraver (page 440).

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.

1.1.47 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 infinity>
  The endpoint of this music. This property is unhappily named in that it does not account for any initial grace notes: the full length of the music is length minus the start time. A value of INF-MOMENT indicates indefinite length.

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.
1.1.48 **LyricEvent**

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

Event classes: lyric-event (page 52), music-event (page 53), rhythmic-event (page 55), and StreamEvent (page 57).

Accepted by: Lyric_engraver (page 431), and Lyric_performer (page 431).

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.

1.1.49 **MeasureCounterEvent**

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

Event classes: measure-counter-event (page 52), music-event (page 53), span-event (page 56), and StreamEvent (page 57).

Accepted by: Measure_counter_engraver (page 433).

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.

1.1.50 **MeasureSpannerEvent**

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

Event classes: measure-spanner-event (page 53), music-event (page 53), span-event (page 56), and StreamEvent (page 57).

Accepted by: Measure_spanner_engraver (page 434).

Properties:

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

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

Articulations on multi-measure rests.

Event classes: multi-measure-articulation-event (page 53), music-event (page 53),
and StreamEvent (page 57).

Accepted by: Multi_measure_rest_engraver (page 436).

Properties:

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

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

1.1.52 MultiMeasureRestEvent

Used internally by MultiMeasureRestMusic to signal rests.

Event classes: general-rest-event (page 51), multi-measure-rest-event (page 53),
music-event (page 53), rhythmic-event (page 55), and StreamEvent (page 57).

Accepted by: Current_chord_text_engraver (page 416), and Multi_measure_rest_engraver (page 436).

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
    general-rest-event
    multi-measure-rest-event)
  The types of this music object; determines by what engraver this music expression is processed.

1.1.53 MultiMeasureRestMusic

Rests that may be compressed into multi-measure 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.

1.1.54 MultiMeasureTextEvent
Texts on multi-measure rests.
   Syntax: R-\markup { \roman "bla" }
   Note the explicit font switch.
   Event classes: multi-measure-text-event (page 53), music-event (page 53), and StreamEvent (page 57).
   Accepted by: Multi_measure_rest_ engraver (page 436).

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.

1.1.55 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.

1.1.56 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 1.1.32 [EventChord], page 12.
   Event classes: melodic-event (page 53), music-event (page 53), note-event (page 54), rhythmic-event (page 55), and StreamEvent (page 57).
Chapter 1: Music definitions

Accepted by: Beat_engraver (page 408), Beat_performer (page 409), Bend_spanner_engraver (page 410), Completion_heads_engraver (page 414), Current_chord_text_engraver (page 416), Drum_note_performer (page 418), Drum_notes_engraver (page 419), Finger_glide_engraver (page 422), Fretboard_engraver (page 423), Note_heads_engraver (page 438), Note_name_engraver (page 438), Note_performer (page 439), Part_combine_engraver (page 441), Phrasing_slur_engraver (page 441), Slur_engraver (page 446), and Tab_note_heads_engraver (page 451).

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.

1.1.57 NoteGroupingEvent

Start or stop grouping brackets.

Event classes: music-event (page 53), note-grouping-event (page 54), and StreamEvent (page 57).

Accepted by: Horizontal_bracket_engraver (page 427).

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.

1.1.58 OttavaEvent

Start or stop an ottava bracket.

Event classes: music-event (page 53), ottava-event (page 54), and StreamEvent (page 57).

Accepted by: Ottava_spanner_engraver (page 439).

Properties:

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

types (list):
   '(ottava-event event)
   The types of this music object; determines by what engraver this music expression is processed.
1.1.59 **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.

1.1.60 **PageBreakEvent**

Allow, forbid or force a page break.

Event classes: break-event (page 49), music-event (page 53), page-break-event (page 54), and StreamEvent (page 57).

Accepted by: Page_turn_engraver (page 440), and Paper_column_engraver (page 440).

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.

1.1.61 **PageTurnEvent**

Allow, forbid or force a page turn.

Event classes: break-event (page 49), music-event (page 53), page-turn-event (page 54), and StreamEvent (page 57).

Accepted by: Page_turn_engraver (page 440), and Paper_column_engraver (page 440).

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.
1.1.62 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.

1.1.63 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.

1.1.64 PercentEvent

Used internally to signal percent repeats.

Event classes: music-event (page 53), percent-event (page 54), and StreamEvent
(page 57).
1.1.65 PercentRepeatedMusic

Repeats encoded by percents and slashes.

Properties:

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

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

- length-callback (procedure):
  ly:calculated-sequential-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:calculated-sequential-music::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.

1.1.66 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: music-event (page 53), pes-or-flexa-event (page 55), and StreamEvent (page 57).

Accepted by: Vaticana_ligature_engraver (page 456).

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.

1.1.67 PhrasingSlurEvent
Start or end phrasing slur.
Syntax: note\( \text{ and } \text{ note}\)
Event classes: music-event (page 53), phrasing-slur-event (page 55), span-event (page 56), and StreamEvent (page 57).
Accepted by: Phrasing_slur_engraver (page 441).
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.

1.1.68 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.

1.1.69 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.

1.1.70 PropertyUnset

Restore the default setting for a context property. See Section 1.1.69 [PropertySet], page 25.

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.

1.1.71 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.
1.1.72 **RehearsalMarkEvent**

Insert a rehearsal mark.

Syntax: `\mark marker`

Example: `\mark 3`

Event classes: `mark-event` (page 52), `music-event` (page 53), `rehearsal-mark-event` (page 55), and `StreamEvent` (page 57).

Accepted by: `Mark_tracking_translator` (page 433).

Properties:

- **name (symbol):** `'RehearsalMarkEvent`
  
  Name of this music object.

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

1.1.73 **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.

1.1.74 **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.

1.1.75 RepeatSlashEvent
Used internally to signal beat repeats.

Event classes: music-event (page 53), repeat-slash-event (page 55), rhythmic-event
(page 55), and StreamEvent (page 57).

Accepted by: Slash_repeat_engraver (page 446).

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.

1.1.76 RepeatTieEvent
Ties for starting a second volta bracket.

Event classes: music-event (page 53), repeat-tie-event (page 55), and StreamEvent
(page 57).

Accepted by: Repeat_tie_engraver (page 444).

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.

1.1.77 RestEvent
A Rest.

Syntax: r4 for a quarter rest.

Event classes: general-rest-event (page 51), music-event (page 53), rest-event
(page 55), rhythmic-event (page 55), and StreamEvent (page 57).
Chapter 1: Music definitions

Accepted by: Completion_rest_engraver (page 414), Current_chord_text_engraver (page 416), Figured_bass_engraver (page 421), and Rest_engraver (page 444).

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
general-rest-event
rest-event)

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

1.1.78 RevertProperty

The opposite of Section 1.1.59 [OverrideProperty], page 22: 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.

1.1.79 ScriptEvent

Add an articulation mark to a note.

Event classes: music-event (page 53), script-event (page 55), and StreamEvent (page 57).

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.
1.1.80 **SectionEvent**

Add a section division, which is typically written as a thin double bar line.

Event classes: `music-event` (page 53), `section-event` (page 55), and `StreamEvent` (page 57).

Accepted by: `Bar_graver` (page 404), and `Divisio_graver` (page 417).

Properties:

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

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

1.1.81 **SectionLabelEvent**

Mark the beginning of a named passage. Does not imply a section division.

Event classes: `music-event` (page 53), `section-label-event` (page 55), and `StreamEvent` (page 57).

Accepted by: `Mark_tracking_translator` (page 433).

Properties:

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

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

1.1.82 **SegnoMarkEvent**

Add a segno mark or bar line.

Event classes: `music-event` (page 53), `segno-mark-event` (page 56), and `StreamEvent` (page 57).

Accepted by: `Bar_graver` (page 404), and `Mark_tracking_translator` (page 433).

Properties:

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

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

Repeats with alternatives placed sequentially and marked with segno, Coda, D.C., etc.

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:calculated-sequential-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):
  - 'SegnoRepeatedMusic
    - Name of this music object.

- start-callback (procedure):
  - ly:calculated-sequential-music::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):
  - '(segno-repeated-music
    - folded-repeated-music
    - repeated-music)
    - The types of this music object; determines by what engraver this music expression is processed.

1.1.84 SequentialAlternativeMusic

Repeat alternatives in sequence.

Syntax: \alternative { alternatives }

Properties:

- elements-callback (procedure):
  - #<procedure 7fca1bfe3ee8 at /build/out/share/lilypond/current/scm/lily/define-music-types.scm:622:30 (m)>
    - Return a list of children, for use by a sequential iterator. Takes a single music parameter.

- iterator-ctor (procedure):
  - ly:alternative-sequence-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):
  'SequentialAlternativeMusic
  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 sequential-alternative-music)
  The types of this music object; determines by what engraver this music expression is
  processed.

1.1.85 SequentialMusic

Music expressions concatenated.

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

Properties:

  elements-callback (procedure):
  #<procedure 7fca1bfe4038 at /build/out/share/lilypond/current/scm/lily/define-music-
  types.scm:635:30 (m)>
  Return a list of children, for use by a sequential iterator. Takes a single music pa-
  rameter.

  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.

1.1.86 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):**
```lisp
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):**
```lisp
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):**
```lisp
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.

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

- **Syntax:** \s4 for a skip equivalent to a quarter rest.
- **Event classes:** music-event (page 53), rhythmic-event (page 55), skip-event (page 56), and StreamEvent (page 57).
- Not accepted by any engraver or performer.

**Properties:**

**iterator-ctor (procedure):**
```lisp
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.

### 1.1.88 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):**
```lisp
ly:simple-music-iterator::constructor
```
Function to construct a music-event-iterator object for this music.
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.

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

Syntax: \skip music

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

1.1.90 SlurEvent
Start or end slur.

Syntax: note( and note)

Event classes: music-event (page 53), slur-event (page 56), span-event (page 56), and StreamEvent (page 57).

Accepted by: Slur_engraver (page 446), and Slur_performer (page 447).

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.
1.1.91 SoloOneEvent

Print ‘Solo 1’.

Event classes: music-event (page 53), part-combine-event (page 54), solo-one-event (page 56), and StreamEvent (page 57).

Accepted by: Part_combine_engraver (page 441).

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.

1.1.92 SoloTwoEvent

Print ‘Solo 2’.

Event classes: music-event (page 53), part-combine-event (page 54), solo-two-event (page 56), and StreamEvent (page 57).

Accepted by: Part_combine_engraver (page 441).

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.

1.1.93 SostenutoEvent

Depress or release sostenuto pedal.

Event classes: music-event (page 53), pedal-event (page 54), sostenuto-event (page 56), span-event (page 56), and StreamEvent (page 57).

Accepted by: Piano_pedal_engraver (page 442), and Piano_pedal_performer (page 443).

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.

1.1.94 SpacingSectionEvent

Start a new spacing section.

  Event classes: music-event (page 53), spacing-section-event (page 56), and StreamEvent (page 57).

  Accepted by: Spacing_engraver (page 447).

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.

1.1.95 SpanEvent

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

  Event classes: music-event (page 53), span-event (page 56), and StreamEvent (page 57).

  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.

1.1.96 StaffHighlightEvent

Start or stop a staff highlight.

  Syntax: \staffHighlight, \stopStaffHighlight.

  Event classes: music-event (page 53), span-event (page 56), staff-highlight-event (page 56), and StreamEvent (page 57).

  Accepted by: Staff_highlight_engraver (page 449).

Properties:

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

  types (list):
    '(staff-highlight-event span-event event)
    The types of this music object; determines by what engraver this music expression is processed.
1.1.97 **StaffSpanEvent**
Start or stop a staff symbol.

Event classes: music-event (page 53), span-event (page 56), staff-span-event (page 57), and StreamEvent (page 57).

Accepted by: Staff_symbol_engraver (page 449).

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.

1.1.98 **StringNumberEvent**
Specify on which string to play this note.

Syntax: \number

Event classes: music-event (page 53), StreamEvent (page 57), and string-number-event (page 57).

Accepted by: Bend_spanner_engraver (page 410), Fretboard_engraver (page 423), and Tab_note_heads_engraver (page 451).

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.

1.1.99 **StrokeFingerEvent**
Specify with which finger to pluck a string.

Syntax: \rightHandFinger text

Event classes: music-event (page 53), StreamEvent (page 57), and stroke-finger-event (page 57).

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.
1.1.100 **SustainEvent**
Depress or release sustain pedal.

Event classes: music-event (page 53), pedal-event (page 54), span-event (page 56), StreamEvent (page 57), and sustain-event (page 57).

Accepted by: Piano_pedal_ engraver (page 442), and Piano_pedal_performer (page 443).

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.

1.1.101 **TempoChangeEvent**
A metronome mark or tempo indication.

Event classes: music-event (page 53), StreamEvent (page 57), and tempo-change-event (page 58).

Accepted by: Metronome_mark_ engraver (page 435).

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.

1.1.102 **TextScriptEvent**
Print text.

Event classes: music-event (page 53), script-event (page 55), StreamEvent (page 57), and text-script-event (page 58).

Accepted by: Text_ engraver (page 452).

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.
1.1.103 **TextSpanEvent**

Start a text spanner, for example, an octavation.

Event classes: music-event (page 53), span-event (page 56), StreamEvent (page 57), and text-span-event (page 58).

Accepted by: Text_spanner_engraver (page 452).

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.

1.1.104 **TieEvent**

A tie.

Syntax: `note-~`

Event classes: music-event (page 53), StreamEvent (page 57), and tie-event (page 58).

Accepted by: Drum_note_performer (page 418), Note_performer (page 439), Tie_engraver (page 452), and Tie_performer (page 453).

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.

1.1.105 **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: musica-transport::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)

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

1.1.106 TimeSignatureEvent

An event created when setting a new time signature

  Event classes: music-event (page 53), StreamEvent (page 57), and time-signature-event (page 58).

  Accepted by: Time_signature_ engraver (page 453), and Time_signature_performer (page 454).

  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.

1.1.107 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.
1.1.108 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.

1.1.109 TremoloEvent

Unmeasured tremolo.

Event classes: music-event (page 53), StreamEvent (page 57), and tremolo-event (page 58).

Accepted by: Stem_engraver (page 450).

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.

1.1.110 TremoloRepeatedMusic

Repeated notes denoted by tremolo beams.

Properties:

- **elements-callback** (procedure):
  - `make-tremolo-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:calculated-sequential-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:calculated-sequential-music::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.

1.1.111 TremoloSpanEvent

Tremolo over two stems.

   Event classes: music-event (page 53), span-event (page 56), StreamEvent (page 57), and
   tremolo-span-event (page 58).

   Accepted by: Chord_tremolo_engraver (page 412).

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.

1.1.112 TrillSpanEvent

Start a trill spanner.

   Event classes: music-event (page 53), span-event (page 56), StreamEvent (page 57), and
   trill-span-event (page 58).

   Accepted by: Trill_spanner_engraver (page 455).

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.
**1.1.113 TupletSpanEvent**

Used internally to signal where tuplet brackets start and stop.

- Event classes: music-event (page 53), span-event (page 56), StreamEvent (page 57), and tuplet-span-event (page 58).
- Accepted by: Stem_engraver (page 450), and Tuplet_engraver (page 455).

**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.

**1.1.114 UnaCordaEvent**

Depress or release una-corda pedal.

- Event classes: music-event (page 53), pedal-event (page 54), span-event (page 56), StreamEvent (page 57), and una-corda-event (page 58).
- Accepted by: Piano_pedal_engraver (page 442), and Piano_pedal_performer (page 443).

**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.

**1.1.115 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:calculated-sequential-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:calculated-sequential-music::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.

1.1.116 UnfoldedSpeccedMusic
Music that appears once repeated music is unfolded.

Properties:

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

length (moment):
  #<Mom 0>
  The endpoint of this music. This property is unhappily named in that it does not
  account for any initial grace notes: the full length of the music is length minus the
  start time. A value of INF-MOMENT indicates indefinite length.

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

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

1.1.117 UnisonoEvent
Print ’a 2’.

Event classes: music-event (page 53), part-combine-event (page 54), StreamEvent
(page 57), and unisono-event (page 59).

Accepted by: Part_combine_engraver (page 441).

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.

1.1.118 UnrelativableMusic
Musical notation 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.

1.1.119 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.
1.1.120 **VoltaRepeatEndEvent**

Signal the end of a volta-style repeat. Multiple end events per start event can be expected when there are alternative endings.

Event classes: `music-event` (page 53), `StreamEvent` (page 57), and `volta-repeat-end-event` (page 59).

Accepted by: `Divisio_engraver` (page 417), `Lyric_repeat_count_engraver` (page 431), `Repeat_acknowledge_engraver` (page 443), and `Signum_repetitionis_engraver` (page 446).

Properties:

- **name** (symbol):
  - 'VoltaRepeatEndEvent
    - Name of this music object.

- **types** (list):
  - '(volta-repeat-end-event event)
    - The types of this music object; determines by what engraver this music expression is processed.

1.1.121 **VoltaRepeatStartEvent**

Signal the start of a volta-style repeat.

Event classes: `music-event` (page 53), `StreamEvent` (page 57), and `volta-repeat-start-event` (page 59).

Accepted by: `Divisio_engraver` (page 417), and `Repeat_acknowledge_engraver` (page 443).

Properties:

- **name** (symbol):
  - 'VoltaRepeatStartEvent
    - Name of this music object.

- **types** (list):
  - '(volta-repeat-start-event event)
    - The types of this music object; determines by what engraver this music expression is processed.

1.1.122 **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:calculated-sequential-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:calculated-sequential-music::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):
   '(
      volta-repeated-music
      folded-repeated-music
      repeated-music
   )

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

1.1.123 VoltaSpanEvent

Used internally to signal where volta brackets start and stop.

   Event classes: music-event (page 53), span-event (page 56), StreamEvent (page 57), and
   volta-span-event (page 59).

   Accepted by: Bar_engraver (page 404), and Volta_engraver (page 456).

   Properties:
      name (symbol):
         'VoltaSpanEvent
         Name of this music object.

      types (list):
         '(
            volta-span-event
            span-event
            event
            post-event
         )

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

1.1.124 VoltaSpeccedMusic

Music for a specific volta within repeated music.

   Properties:
      iterator-ctor (procedure):
         ly:volta-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):
         'VoltaSpeccedMusic
         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.
Chapter 1: Music definitions

1.1.125 VowelTransitionEvent
A vowel transition between lyric syllables.

Event classes: music-event (page 53), StreamEvent (page 57), and vowel-transition-event (page 59).

Accepted by: Hyphen_engraver (page 427).

Properties:
- name (symbol): 'VowelTransitionEvent
  Name of this music object.
- types (list):
  '(post-event vowel-transition-event event)
  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 AbsoluteDynamicEvent (page 1).

Accepted by: Dynamic_engraver (page 420), and Dynamic_performer (page 420).

1.2.2 ad-hoc-jump-event
Music event type ad-hoc-jump-event is in music objects of type AdHocJumpEvent (page 1).

Accepted by: Bar_engraver (page 404), and Jump_engraver (page 428).

1.2.3 ad-hoc-mark-event
Music event type ad-hoc-mark-event is in music objects of type AdHocMarkEvent (page 1).

Accepted by: Mark_tracking_translator (page 433).

1.2.4 alternative-event
Music event type alternative-event is in music objects of type AlternativeEvent (page 2).

Accepted by: Timing_translator (page 454).

1.2.5 annotate-output-event
Music event type annotate-output-event is in music objects of type AnnotateOutputEvent (page 2).

Accepted by: Balloon_engraver (page 404).

1.2.6 apply-output-event
Music event type apply-output-event is in music objects of type ApplyOutputEvent (page 3).

Accepted by: Output_property_engraver (page 439).
1.2.7 arpeggio-event
Music event type arpeggio-event is in music objects of type ArpeggioEvent (page 3).
   Accepted by: Arpeggio_engraver (page 402).

1.2.8 articulation-event
Music event type articulation-event is in music objects of type ArticulationEvent (page 3).
   Accepted by: Beat_engraver (page 408), Beat_performer (page 409), Drum_note_performer (page 418), Note_performer (page 439), and Script_engraver (page 445).

1.2.9 bar-event
Music event type bar-event is in music objects of type BarEvent (page 4).
   Accepted by: Timing_translator (page 454).

1.2.10 bass-figure-event
Music event type bass-figure-event is in music objects of type BassFigureEvent (page 5).
   Accepted by: Figured_bass_engraver (page 421).

1.2.11 beam-event
Music event type beam-event is in music objects of type BeamEvent (page 5).
   Accepted by: Beam_engraver (page 407), Beam_performer (page 408), and Grace_beam_engraver (page 425).

1.2.12 beam-forbid-event
Music event type beam-forbid-event is in music objects of type BeamForbidEvent (page 5).
   Accepted by: Auto_beam_engraver (page 402), and Grace_auto_beam_engraver (page 424).

1.2.13 bend-after-event
Music event type bend-after-event is in music objects of type BendAfterEvent (page 6).
   Accepted by: Bend_engraver (page 409).

1.2.14 bend-span-event
Music event type bend-span-event is in music objects of type BendSpanEvent (page 6).
   Accepted by: Bend_spanner_engraver (page 410).

1.2.15 break-dynamic-span-event
Music event type break-dynamic-span-event is in music objects of type BreakDynamicSpanEvent (page 6).
   Accepted by: Dynamic_engraver (page 420).

1.2.16 break-event
Music event type break-event is in music objects of type LineBreakEvent (page 17), PageBreakEvent (page 22), and PageTurnEvent (page 22).
   Accepted by: Page_turn_engraver (page 440), and Paper_column_engraver (page 440).
1.2.17 break-span-event
Music event type break-span-event is in music objects of type BreakDynamicSpanEvent (page 6).
Not accepted by any engraver or performer.

1.2.18 breathing-event
Music event type breathing-event is in music objects of type BreathingEvent (page 7).
Accepted by: Breathing_sign_engraver (page 410), and Note_performer (page 439).

1.2.19 caesura-event
Music event type caesura-event is in music objects of type CaesuraEvent (page 7).
Accepted by: Bar_engraver (page 404), Caesura_engraver (page 410), and Divisio_engraver (page 417).

1.2.20 cluster-note-event
Music event type cluster-note-event is in music objects of type ClusterNoteEvent (page 7).
Accepted by: Cluster_spanner_engraver (page 413).

1.2.21 coda-mark-event
Music event type coda-mark-event is in music objects of type CodaMarkEvent (page 8).
Accepted by: Bar_engraver (page 404), and Mark_tracking_translator (page 433).

1.2.22 completize-extender-event
Music event type completize-extender-event is in music objects of type CompletizeExtenderEvent (page 8).
Accepted by: Extender_engraver (page 421).

1.2.23 crescendo-event
Music event type crescendo-event is in music objects of type CrescendoEvent (page 9).
Accepted by: Dynamic_performer (page 420).

1.2.24 dal-segno-event
Music event type dal-segno-event is in music objects of type DalSegnoEvent (page 10).
Accepted by: Bar_engraver (page 404), Jump_engraver (page 428), and Volta_engraver (page 456).

1.2.25 decrescendo-event
Music event type decrescendo-event is in music objects of type DecrescendoEvent (page 10).
Accepted by: Dynamic_performer (page 420).

1.2.26 double-percent-event
Music event type double-percent-event is in music objects of type DoublePercentEvent (page 10).
Accepted by: Double_percent_repeat_engraver (page 418).
1.2.27 duration-line-event
Music event type duration-line-event is in music objects of type DurationLineEvent (page 11).
   Accepted by: Duration_line_engraver (page 419).

1.2.28 dynamic-event
Music event type dynamic-event is in music objects of type AbsoluteDynamicEvent (page 1).
   Not accepted by any engraver or performer.

1.2.29 episema-event
Music event type episema-event is in music objects of type EpisemaEvent (page 11).
   Accepted by: Episema_engraver (page 421).

1.2.30 extender-event
Music event type extender-event is in music objects of type ExtenderEvent (page 12).
   Accepted by: Extender_engraver (page 421).

1.2.31 fine-event
Music event type fine-event is in music objects of type FineEvent (page 13).
   Accepted by: Bar_engraver (page 404), Divisio_engraver (page 417), Jump_engraver (page 428), Timing_translator (page 454), and Volta_engraver (page 456).

1.2.32 finger-glide-event
Music event type finger-glide-event is in music objects of type FingerGlideEvent (page 13).
   Not accepted by any engraver or performer.

1.2.33 fingering-event
Music event type fingering-event is in music objects of type FingeringEvent (page 13).
   Accepted by: Fingering_engraver (page 422), Fretboard_engraver (page 423), and Tab_note_heads_engraver (page 451).

1.2.34 footnote-event
Music event type footnote-event is in music objects of type FootnoteEvent (page 14).
   Not accepted by any engraver or performer.

1.2.35 general-rest-event
Music event type general-rest-event is in music objects of type MultiMeasureRestEvent (page 19), and RestEvent (page 28).
   Accepted by: Current_chord_text_engraver (page 416).

1.2.36 glissando-event
Music event type glissando-event is in music objects of type GlissandoEvent (page 14).
   Accepted by: Glissando_engraver (page 424).

1.2.37 harmonic-event
Music event type harmonic-event is in music objects of type HarmonicEvent (page 15).
   Not accepted by any engraver or performer.
1.2.38 **hyphen-event**
Music event type hyphen-event is in music objects of type HyphenEvent (page 15).
   Accepted by: Hyphen_engraver (page 427).

1.2.39 **key-change-event**
Music event type key-change-event is in music objects of type KeyChangeEvent (page 15).
   Accepted by: Key_engraver (page 429), and Key_performer (page 430).

1.2.40 **label-event**
Music event type label-event is in music objects of type LabelEvent (page 16).
   Accepted by: Paper_column_engraver (page 440).

1.2.41 **laissez-vibrer-event**
Music event type laissez-vibrer-event is in music objects of type LaissezVibrerEvent (page 16).
   Accepted by: Laissez_vibrer_engraver (page 430).

1.2.42 **layout-instruction-event**
Music event type layout-instruction-event is in music objects of type ApplyOutputEvent (page 3).
   Not accepted by any engraver or performer.

1.2.43 **ligature-event**
Music event type ligature-event is in music objects of type LigatureEvent (page 16).
   Accepted by: Kievan_ligature_engraver (page 430), Ligature_bracket_engraver (page 431), Mensural_ligature_engraver (page 435), and Vaticana_ligature_engraver (page 456).

1.2.44 **line-break-event**
Music event type line-break-event is in music objects of type LineBreakEvent (page 17).
   Not accepted by any engraver or performer.

1.2.45 **lyric-event**
Music event type lyric-event is in music objects of type LyricEvent (page 18).
   Accepted by: Lyric_engraver (page 431), and Lyric_performer (page 431).

1.2.46 **mark-event**
Music event type mark-event is in music objects of type AdHocMarkEvent (page 1), and RehearsalMarkEvent (page 27).
   Not accepted by any engraver or performer.

1.2.47 **measure-counter-event**
Music event type measure-counter-event is in music objects of type MeasureCounterEvent (page 18).
   Accepted by: Measure_counter_engraver (page 433).
1.2.48 measure-spanner-event
Music event type measure-spanner-event is in music objects of type MeasureSpannerEvent (page 18).
Accepted by: Measure_spanner_engraver (page 434).

1.2.49 melodic-event
Music event type melodic-event is in music objects of type ClusterNoteEvent (page 7), and NoteEvent (page 20).
Not accepted by any engraver or performer.

1.2.50 multi-measure-articulation-event
Music event type multi-measure-articulation-event is in music objects of type MultiMeasureArticulationEvent (page 19).
Accepted by: Multi_measure_rest_engraver (page 436).

1.2.51 multi-measure-rest-event
Music event type multi-measure-rest-event is in music objects of type MultiMeasureRestEvent (page 19).
Accepted by: Multi_measure_rest_engraver (page 436).

1.2.52 multi-measure-text-event
Music event type multi-measure-text-event is in music objects of type MultiMeasureTextEvent (page 20).
Accepted by: Multi_measure_rest_engraver (page 436).

1.2.53 music-event
Music event type music-event is in music objects of type AbsoluteDynamicEvent (page 1), AdHocJumpEvent (page 1), AdHocMarkEvent (page 1), AlternativeEvent (page 2), AnnotateOutputEvent (page 2), ApplyOutputEvent (page 3), ArpeggioEvent (page 3), ArticulationEvent (page 3), BarEvent (page 4), BassFigureEvent (page 5), BeamEvent (page 5), BeamForbidEvent (page 5), BendAfterEvent (page 6), BendSpanEvent (page 6), BreakDynamicSpanEvent (page 6), BreathingEvent (page 7), CaesuraEvent (page 7), ClusterNoteEvent (page 7), CodaMarkEvent (page 8), CompleteExtenderEvent (page 8), CrescendoEvent (page 9), DalSegnoEvent (page 10), DecrescendoEvent (page 10), DoublePercentEvent (page 10), DurationLineEvent (page 11), EpisemaEvent (page 11), ExtenderEvent (page 12), FineEvent (page 13), FingerGlideEvent (page 13), FingeringEvent (page 13), FootnoteEvent (page 14), GlissandoEvent (page 14), HarmonicEvent (page 15), HyphenEvent (page 15), KeyChangeEvent (page 15), LabelEvent (page 16), LaissezVibrerEvent (page 16), LigatureEvent (page 16), LineBreakEvent (page 17), LyricEvent (page 18), MeasureCounterEvent (page 18), MeasureSpannerEvent (page 18), MultiMeasureArticulationEvent (page 19), MultiMeasureRestEvent (page 19), MultiMeasureTextEvent (page 20), NoteEvent (page 20), NoteGroupingEvent (page 21), OttavinaEvent (page 21), PageBreakEvent (page 22), PageTurnEvent (page 22), PercentEvent (page 23), PesOrFlexaEvent (page 24), PhrasingSlurEvent (page 25), RehearsalMarkEvent (page 27), RepeatSlashEvent (page 28), RepeatTieEvent (page 28), RestEvent (page 28), ScriptEvent (page 29), SectionEvent (page 30), SectionLabelEvent (page 30), SegnoMarkEvent (page 30), SkipEvent (page 33), SlurEvent (page 34), SoloOneEvent (page 35), SoloTwoEvent (page 35), SostenutoEvent (page 35), SpacingSectionEvent (page 36), SpanEvent (page 36), StaffHighlightEvent (page 36), StaffSpanEvent (page 37), StringNumberEvent (page 37), StrokeFingerEvent (page 37), SustainEvent
1.2.54 note-event
Music event type note-event is in music objects of type NoteEvent (page 20).

Accepted by: Beat_engraver (page 408), Beat_performer (page 409), Bend_spanner_engraver (page 410), Completion_heads_engraver (page 414), Current_chord_text_engraver (page 416), Drum_note_performer (page 418), Drum_notes_engraver (page 419), Finger_glide_engraver (page 422), Fretboard_engraver (page 423), Note_heads_engraver (page 438), Note_name_engraver (page 438), Note_performer (page 439), Part_combine_engraver (page 441), Phrasing_slur_engraver (page 441), Slur_engraver (page 446), and Tab_note_heads_engraver (page 451).

1.2.55 note-grouping-event
Music event type note-grouping-event is in music objects of type NoteGroupingEvent (page 21).

Accepted by: Horizontal_bracket_engraver (page 427).

1.2.56 ottava-event
Music event type ottava-event is in music objects of type OttavaEvent (page 21).

Accepted by: Ottava_spanner_engraver (page 439).

1.2.57 page-break-event
Music event type page-break-event is in music objects of type PageBreakEvent (page 22).

Not accepted by any engraver or performer.

1.2.58 page-turn-event
Music event type page-turn-event is in music objects of type PageTurnEvent (page 22).

Not accepted by any engraver or performer.

1.2.59 part-combine-event
Music event type part-combine-event is in music objects of type SoloOneEvent (page 35), SoloTwoEvent (page 35), and UnisonoEvent (page 44).

Accepted by: Part_combine_engraver (page 441).

1.2.60 pedal-event
Music event type pedal-event is in music objects of type SostenutoEvent (page 35), SustainEvent (page 38), and UnaCordaEvent (page 43).

Not accepted by any engraver or performer.

1.2.61 percent-event
Music event type percent-event is in music objects of type PercentEvent (page 23).

Accepted by: Percent_repeat_engraver (page 441).
1.2.62 pes-or-flexa-event
Music event type pes-or-flexa-event is in music objects of type PesOrFlexaEvent (page 24).
   Accepted by: Vaticana_ligature_engraver (page 456).

1.2.63 phrasing-slur-event
Music event type phrasing-slur-event is in music objects of type PhrasingSlurEvent (page 25).
   Accepted by: Phrasing_slur_engraver (page 441).

1.2.64 rehearsal-mark-event
Music event type rehearsal-mark-event is in music objects of type RehearsalMarkEvent (page 27).
   Accepted by: Mark_tracking_translator (page 433).

1.2.65 repeat-slash-event
Music event type repeat-slash-event is in music objects of type RepeatSlashEvent (page 28).
   Accepted by: Slash_repeat_engraver (page 446).

1.2.66 repeat-tie-event
Music event type repeat-tie-event is in music objects of type RepeatTieEvent (page 28).
   Accepted by: Repeat_tie_engraver (page 444).

1.2.67 rest-event
Music event type rest-event is in music objects of type RestEvent (page 28).
   Accepted by: Completion_rest_engraver (page 414), Figured_bass_engraver (page 421), and Rest_engraver (page 444).

1.2.68 rhythmic-event
Music event type rhythmic-event is in music objects of type BassFigureEvent (page 5),
   ClusterNoteEvent (page 7), DoublePercentEvent (page 10), LyricEvent (page 18),
   MultiMeasureRestEvent (page 19), NoteEvent (page 20), RepeatSlashEvent (page 28),
   RestEvent (page 28), and SkipEvent (page 33).
   Not accepted by any engraver or performer.

1.2.69 script-event
Music event type script-event is in music objects of type ArticulationEvent (page 3),
   ScriptEvent (page 29), and TextScriptEvent (page 38).
   Not accepted by any engraver or performer.

1.2.70 section-event
Music event type section-event is in music objects of type SectionEvent (page 30).
   Accepted by: Bar_engraver (page 404), and Divisio_engraver (page 417).

1.2.71 section-label-event
Music event type section-label-event is in music objects of type SectionLabelEvent (page 30).
   Accepted by: Mark_tracking_translator (page 433).
1.2.72 **segno-mark-event**
Music event type `segno-mark-event` is in music objects of type `SegnoMarkEvent` (page 30).
   Accepted by: `Bar_engraver` (page 404), and `Mark_tracking_translator` (page 433).

1.2.73 **skip-event**
Music event type `skip-event` is in music objects of type `SkipEvent` (page 33).
   Not accepted by any engraver or performer.

1.2.74 **slur-event**
Music event type `slur-event` is in music objects of type `SlurEvent` (page 34).
   Accepted by: `Slur_engraver` (page 446), and `Slur_performer` (page 447).

1.2.75 **solo-one-event**
Music event type `solo-one-event` is in music objects of type `SoloOneEvent` (page 35).
   Not accepted by any engraver or performer.

1.2.76 **solo-two-event**
Music event type `solo-two-event` is in music objects of type `SoloTwoEvent` (page 35).
   Not accepted by any engraver or performer.

1.2.77 **sostenuto-event**
Music event type `sostenuto-event` is in music objects of type `SostenutoEvent` (page 35).
   Accepted by: `Piano_pedal_engraver` (page 442), and `Piano_pedal_performer` (page 443).

1.2.78 **spacing-section-event**
Music event type `spacing-section-event` is in music objects of type `SpacingSectionEvent` (page 36).
   Accepted by: `Spacing_engraver` (page 447).

1.2.79 **span-dynamic-event**
Music event type `span-dynamic-event` is in music objects of type `CrescendoEvent` (page 9),
   and `DecrescendoEvent` (page 10).
   Accepted by: `Dynamic_engraver` (page 420).

1.2.80 **span-event**
Music event type `span-event` is in music objects of type `BeamEvent` (page 5),
   `BendSpanEvent` (page 6), `CrescendoEvent` (page 9), `DecrescendoEvent` (page 10), `EpisemaEvent`
   (page 11), `FingerGlideEvent` (page 13), `LigatureEvent` (page 16), `MeasureCounterEvent`
   (page 18), `MeasureSpannerEvent` (page 18), `PhrasingSlurEvent` (page 25), `SlurEvent`
   (page 34), `SostenutoEvent` (page 35), `SpanEvent` (page 36), `StaffHighlightEvent`
   (page 36), `StaffSpanEvent` (page 37), `SustainEvent` (page 38), `TextSpanEvent` (page 39),
   `TremoloSpanEvent` (page 42), `TrillSpanEvent` (page 42), `TupletSpanEvent` (page 43),
   `UnaCordaEvent` (page 43), and `VoltaSpanEvent` (page 47).
   Not accepted by any engraver or performer.

1.2.81 **staff-highlight-event**
Music event type `staff-highlight-event` is in music objects of type `StaffHighlightEvent` (page 36).
   Accepted by: `Staff_highlight_engraver` (page 449).
1.2.82 **staff-span-event**

Music event type **staff-span-event** is in music objects of type StaffSpanEvent (page 37).

Accepted by: Staff_symbol_engraver (page 449).

1.2.83 **StreamEvent**

Music event type **StreamEvent** is in music objects of type AbsoluteDynamicEvent (page 1), AdHocMarkEvent (page 1), AdHocMarkEvent (page 1), AlternativeEvent (page 2), AnnotateOutputEvent (page 2), ApplyOutputEvent (page 3), ArticulationEvent (page 3), BarEvent (page 4), BassFigureEvent (page 5), BeamEvent (page 5), BeamForbidEvent (page 5), BendAfterEvent (page 6), BendSpanEvent (page 6), BreakDynamicSpanEvent (page 6), BreathingEvent (page 7), CaesuraEvent (page 7), ClusterNoteEvent (page 7), CodaMarkEvent (page 8), CompletizeExtenderEvent (page 8), CrescendoEvent (page 9), DalSegnoEvent (page 10), DecrescendoEvent (page 10), DoublePercentEvent (page 10), DurationLineEvent (page 11), EpisemaEvent (page 11), ExtenderEvent (page 12), FineEvent (page 13), FingerGlideEvent (page 13), FingeringEvent (page 13), FootnoteEvent (page 14), GlissandoEvent (page 14), HarmonicEvent (page 15), HyphenEvent (page 15), KeyChangeEvent (page 15), LabelEvent (page 16), LaissezVibrerEvent (page 16), LigatureEvent (page 16), LineBreakEvent (page 17), LyricEvent (page 18), MeasureCounterEvent (page 18), MeasureExtenderEvent (page 18), MultiMeasureTextEvent (page 19), MultiMeasureArticulationEvent (page 19), MultiMeasureArticulationEvent (page 19), MultiMeasureExtenderEvent (page 19), NoteEvent (page 20), NoteGroupingEvent (page 21), OttavaEvent (page 21), PageBreakEvent (page 22), PageTurnEvent (page 22), PercentEvent (page 23), PesOrFlexaEvent (page 24), PhrasingSlurEvent (page 25), RehearsalMarkEvent (page 27), RepeatSlashEvent (page 28), RepeatTieEvent (page 28), RestEvent (page 28), ScriptEvent (page 29), SectionEvent (page 30), SectionLabelEvent (page 30), SegnoMarkEvent (page 30), SkipEvent (page 33), SlurEvent (page 34), SoloOneEvent (page 35), SoloTwoEvent (page 35), SostenutoEvent (page 35), SpacingSectionEvent (page 36), SpanEvent (page 36), StaffHighlightEvent (page 36), StaffSpanEvent (page 37), StringNumberEvent (page 37), StrokeFingerEvent (page 37), SustainEvent (page 38), TempoChangeEvent (page 38), TextScriptEvent (page 38), TextSpanEvent (page 39), TieEvent (page 39), TimeSignatureEvent (page 40), TremoloEvent (page 41), TrilloSpanEvent (page 42), TrillSpanEvent (page 42), TupletSpanEvent (page 43), UnaCordaEvent (page 43), UnisonoEvent (page 44), VoltaRepeatEndEvent (page 46), VoltaRepeatStartEvent (page 46), VoltaSpanEvent (page 47), and VowelTransitionEvent (page 48).

Not accepted by any engraver or performer.

1.2.84 **string-number-event**

Music event type **string-number-event** is in music objects of type StringNumberEvent (page 37).

Accepted by: Bend_spanner_engraver (page 410), Fretboard_engraver (page 423), and Tab_note_heads_engraver (page 451).

1.2.85 **stroke-finger-event**

Music event type **stroke-finger-event** is in music objects of type StrokeFingerEvent (page 37).

Not accepted by any engraver or performer.

1.2.86 **sustain-event**

Music event type **sustain-event** is in music objects of type SustainEvent (page 38).
1.2.87 tempo-change-event
Music event type tempo-change-event is in music objects of type TempoChangeEvent (page 38).
   Accepted by: Metronome_mark_engraver (page 435).

1.2.88 text-script-event
Music event type text-script-event is in music objects of type TextScriptEvent (page 38).
   Accepted by: Text_engraver (page 452).

1.2.89 text-span-event
Music event type text-span-event is in music objects of type TextSpanEvent (page 39).
   Accepted by: Text_spanner_engraver (page 452).

1.2.90 tie-event
Music event type tie-event is in music objects of type TieEvent (page 39).
   Accepted by: Drum_note_performer (page 418), Note_performer (page 439), Tie_engraver (page 452), and Tie_performer (page 453).

1.2.91 time-signature-event
Music event type time-signature-event is in music objects of type TimeSignatureEvent (page 40).
   Accepted by: Time_signature_engraver (page 453), and Time_signature_performer (page 454).

1.2.92 tremolo-event
Music event type tremolo-event is in music objects of type TremoloEvent (page 41).
   Accepted by: Stem_engraver (page 450).

1.2.93 tremolo-span-event
Music event type tremolo-span-event is in music objects of type TremoloSpanEvent (page 42).
   Accepted by: Chord_tremolo_engraver (page 412).

1.2.94 trill-span-event
Music event type trill-span-event is in music objects of type TrillSpanEvent (page 42).
   Accepted by: Trill_spanner_engraver (page 455).

1.2.95 tuplet-span-event
Music event type tuplet-span-event is in music objects of type TupletSpanEvent (page 43).
   Accepted by: Stem_engraver (page 450), and Tuplet_engraver (page 455).

1.2.96 una-corda-event
Music event type una-corda-event is in music objects of type UnaCordaEvent (page 43).
   Accepted by: Piano_pedal_engraver (page 442), and Piano_pedal_performer (page 443).
1.2.97 *unisono-event*

Music event type *unisono-event* is in music objects of type *UnisonoEvent* (page 44).

Not accepted by any engraver or performer.

1.2.98 *volta-repeat-end-event*

Music event type *volta-repeat-end-event* is in music objects of type *VoltaRepeatEndEvent* (page 46).

Accepted by: *Divisio_engraver* (page 417), *Lyric_repeat_count_engraver* (page 431), *Repeat_acknowledge_engraver* (page 443), and *Signum_repetitionis_engraver* (page 446).

1.2.99 *volta-repeat-start-event*

Music event type *volta-repeat-start-event* is in music objects of type *VoltaRepeatStartEvent* (page 46).

Accepted by: *Divisio_engraver* (page 417), and *Repeat_acknowledge_engraver* (page 443).

1.2.100 *volta-span-event*

Music event type *volta-span-event* is in music objects of type *VoltaSpanEvent* (page 47).

Accepted by: *Bar_engraver* (page 404), and *Volta_engraver* (page 456).

1.2.101 *vowel-transition-event*

Music event type *vowel-transition-event* is in music objects of type *VowelTransitionEvent* (page 48).

Accepted by: *Hyphen_engraver* (page 427).

1.3 Music properties

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

- **alteration** (number)
  - Alteration for figured bass.

- **alteration-bracket** (boolean)
  - Put brackets around bass figure alteration.

- **alternative-dir** (direction)
  - Indicates that an alternative-event is the first (-1), middle (0), or last (1) of group of alternate endings.

- **alternative-number** (non-negative, exact integer)
  - The index of the current alternative element, starting from one.

- **articulation-type** (symbol)
  - Key for script definitions alist.

- **articulations** (list of music objects)
  - Articulation events specifically for this note.

- **associated-context** (string)
  - Name of the context associated with this \lyricstos section.

- **associated-context-type** (symbol)
  - Type of the context associated with this \lyricstos 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.

bar-type (string)
    The type of bar line to create, e.g., "|

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.

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

change-tag (symbol)
    Tag identifying the musical scope of a context change. The change applies to the nearest enclosing music with this tag.

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.

color (color)
    The color of a highlight.

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

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.

digit (non-negative, exact 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.

fine-folded (boolean)
   True in a fine-event that is issued from within a folded repeat (segno or volta).

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 (non-negative, exact integer)
   Sequence number of a mark. 1 is first.

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

length (moment)
   The endpoint of this music. This property is unhappily named in that it does not account
   for any initial grace notes: the full length of the music is length minus the start time. A
   value of INF-MOMENT indicates indefinite length.

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.

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-body-start-moment (moment)
In a D.S. event, the moment of the segno.

repeat-count (non-negative, exact integer)
The number of times to perform a \repeat.
return-count (non-negative, exact integer)
The number of times to perform a D.S.

search-direction (direction)
Limits the scope of \context searches.

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-depth (non-negative, exact integer)
   The depth in the repeat structure.

volta-numbers (number list)
   Volte to which this music applies.

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): Arpeggio (page 483), InstrumentName (page 559), SpanBarStub (page 628), StaffGrouper (page 630), SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), and VerticalAlignment (page 668).

This context sets the following properties:

- Revert grob property extra-spacing-width in DynamicText (page 539),
- Set context property instrumentName to '('.
- Set context property localAlterations to #f.
- Set context property localAlterations to '.(.'.
- Set context property shortInstrumentName to '('.
- Set context property systemStartDelimiter to 'SystemStartBracket'.
- Set context property topLevelAlignment to #f.
- Set grob property extra-spacing-width in DynamicText (page 539), to #f.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Staff (page 287).

Context ChoirStaff can contain ChoirStaff (page 66), ChordNames (page 95), Devnull (page 108), DrumStaff (page 108), Dynamics (page 120), FiguredBass (page 132), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

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

Instrument_name engraver (page 427)
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 items (clef, key signature, etc.).

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): InstrumentName (page 559).

Output_property_engraver (page 439)
    Apply a procedure to any grob acknowledged.
    Music types accepted: apply-output-event (page 48),

Span_arpeggio_engraver (page 447)
    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): Arpeggio (page 483).

Span_bar_stub_engraver (page 448)
    Make stubs for span bars in all contexts that the span bars cross.
    This engraver creates the following layout object(s): SpanBarStub (page 628).

System_start_delimiter_engraver (page 450)
    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 items (clef, key signature, etc.).
        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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

Vertical_align_engraver (page 456)
    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): StaffGrouper (page 630), and VerticalAlignment (page 668).
2.1.2 ChordGrid

Creates chord grid notation. This context is always part of a ChordGridScore context.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): BarLine (page 486), ChordSquare (page 509), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), GridChordName (page 553), PercentRepeat (page 602), PercentRepeatCounter (page 603), StaffSymbol (page 632), SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), and VerticalAxisGroup (page 668).

This context sets the following properties:

• Set grob property `font-size` in BarLine (page 486), to 3.
• Set grob property `hair-thickness` in BarLine (page 486), to 2.
• Set grob property `kern` in BarLine (page 486), to 5.
• Set grob property `line-positions` in StaffSymbol (page 632), to `(−13.5 13.5)`
• Set grob property `thickness` in StaffSymbol (page 632), to 2.
• Set grob property `thickness` in SystemStartBar (page 644), to 2.

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):

Alteration_glyph_engraver (page 401)

Set the `glyph-name-alist` of all grobs having the `accidental-switch-interface` to the value of the context’s `alterationGlyphs` property, when defined.

Properties (read)

alterationGlyphs (list)

Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group_engraver (page 403)

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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).
Bar_engraver (page 404)

Create bar lines for various commands, including \bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59),

Properties (read)

caesuraType (list)
An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

The first argument is the context.

The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:..’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.|:’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘.|.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘|.|S’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.
fineSegnoBarType (string)
  Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|S’.

fineStartRepeatSegnoBarType (string)
  Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|S.|:’.

forbidBreakBetweenBarLines (boolean)
  If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
  Bar line to insert at a measure boundary.

repeatCommands (list)
  A list of commands related to volta-style repeats. In general, each element is a list, '(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

  end-repeat
    End a repeated section.

  start-repeat
    Start a repeated section.

  volta text
    If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
  Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
  Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
  A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
  Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
  Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
  Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
  The current bar line type, or ‘() if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.
Properties (write)

\texttt{currentBarLine} (graphical (layout) object)
Set to the BarLine that \texttt{Bar_engraver} has created in the current timestep.

\texttt{forbidBreak} (boolean)
If set to \#t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): \texttt{BarLine} (page 486).

\texttt{Chord_square_engraver} (page 412)
Engrave chord squares in chord grids.
Properties (read)

\texttt{currentCommandColumn} (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).

This engraver creates the following layout object(s): \texttt{ChordSquare} (page 509).

\texttt{Current_chord_text_engraver} (page 416)
Catch note and rest events and generate the appropriate chord text using \texttt{chordNameFunction}. Actually creating a chord name grob is left to other engravers.
Music types accepted: \texttt{general-rest-event} (page 51), and \texttt{note-event} (page 54),
Properties (read)

\texttt{chordNameExceptions} (list)
An alist of chord exceptions. Contains (\texttt{chord}, \texttt{markup}) entries.

\texttt{chordNameFunction} (procedure)
The function that converts lists of pitches to chord names.

\texttt{chordNoteNamer} (procedure)
A function that converts from a pitch object to a text markup. Used for single pitches.

\texttt{chordRootNamer} (procedure)
A function that converts from a pitch object to a text markup. Used for chords.

\texttt{majorSevenSymbol} (markup)
How should the major 7th be formatted in a chord name?

\texttt{noChordSymbol} (markup)
Markup to be displayed for rests in a ChordNames context.

Properties (write)

\texttt{currentChordCause} (stream event)
Event cause of the chord that should be created in this time step (if any).

\texttt{currentChordText} (markup)
In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

\texttt{Double_percent_repeat_engraver} (page 418)
Make double measure repeats.
Music types accepted: \texttt{double-percent-event} (page 50),
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, except if explicitly requested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

Grid_chord_name_engraver (page 425)
Read **currentChordText** to create chord names adapted for typesetting within a chord grid.

Properties (read)

**currentChordCause** (stream event)
Event cause of the chord that should be created in this time step (if any).

**currentChordText** (markup)
In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

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

This engraver creates the following layout object(s): GridChordName (page 553).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),

Properties (read)

**countPercentRepeats** (boolean)
If set, produce counters for percent repeats.

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

**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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

**Staff_symbol_engraver** (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

**System_start_delimiter_engraver** (page 450)
Create a system start delimiter (i.e., a SystemStartBar, SystemStartBracket, SystemStartBracket or SystemStartSquare spanner).

Properties (read)
- `currentCommandColumn` (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
- `systemStartDelimiter` (symbol)
  Which grob to make for the start of the system/staff? Set to SystemStartBar, SystemStartBracket or SystemStartBar.
- `systemStartDelimiterHierarchy` (pair)
  A nested list, indicating the nesting of a start delimiters.

This engraver creates the following layout object(s): SystemStartBar (page 644), SystemStartBar (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

### 2.1.3 ChordGridScore
Top-level context replacing Score in chord grid notation. Compared to Score, it uses proportional notation, and has a few other settings like removing bar numbers.

This context also accepts commands for the following context(s): Score (page 263), and Timing (page 263).

This context creates the following layout object(s): BreakAlignGroup (page 500), BreakAlignment (page 501), CenteredBarNumberLineSpanner (page 507), CodaMark (page 515), ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), GraceSpacing (page 553), JumpScript (page 561), LeftEdge (page 571), MetronomeMark (page 585), NonMusicalPaperColumn (page 593), PaperColumn (page 600), Parentheses (page 601), RehearsalMark (page 607), SectionLabel (page 615), SegnoMark (page 616), SpacingSpanner (page 626), StaffGrouper (page 630), VerticalAlignment (page 668), VoltaBracket (page 671), and VoltaBracketSpanner (page 672).

This context sets the following properties:
- Set context property `additionalPitchPrefix` to "".
- Set context property `aDueText` to "a2".
- Set context property `alterationGlyphs` to #f.
- Set context property `alternativeRestores` to:
  "(measurePosition measureLength lastChord)
- Set context property `associatedVoiceType` to 'Voice.
- Set context property `autoAccidentals` to:
  "(Staff #<procedure 5575a375c520 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
- Set context property `autoBeamCheck` to default-auto-beam-check.
- Set context property `autoBeaming` to #t.
• Set context property autoCautionaries to '().
• Set context property barCheckSynchronize to #f.
• Set context property barNumberFormatter to robust-bar-number-function.
• Set context property barNumberVisibility to first-bar-number-invisible-and-no-parenthesized-bar-numbers.
• Set context property beamHalfMeasure to #t.
• Set context property breathMarkDefinitions to:
  '((caesura
    (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.caesura.straight")
    (chantdoublebar
     (extra-spacing-width -1.0 . 0.0)
     (stencil
      .
      #<procedure ly:breathing-sign::finalis (_)>)
     (Y-offset . 0.0))
    (chantfullbar
     (extra-spacing-width -1.0 . 0.0)
     (stencil
      .
      #<procedure ly:breathing-sign::divisio-maxima (_)>)
     (Y-offset . 0.0))
    (chanthalfbar
     (extra-spacing-height
      .
      #<procedure item::extra-spacing-height-including-staff (grob)>)
     (extra-spacing-width -1.0 . 0.0)
     (stencil
      .
      #<procedure ly:breathing-sign::divisio-maior (_)>)
     (Y-offset . 0.0))
    (chantquarterbar
     (extra-spacing-height
      .
      #<procedure item::extra-spacing-height-including-staff (grob)>)
     (extra-spacing-width -1.0 . 0.0)
     (stencil
      .
      #<procedure ly:breathing-sign::divisio-minima (_)>))
    (comma (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.rcomma")
    (curvedcaesura
     (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.caesura.curved")
    (outsidecomma
     (outside-staff-priority . 40)
     (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.rcomma")
    (spacer
     (text #<procedure null-markup (layout props)>)
    (tickmark
Set context property `breathMarkType` to 'comma.'

Set context property `caesuraType` to: '((breath . caesura))'

Set context property `centerBarNumbers` to #f.

Set context property `chordNameExceptions` to:

`'((#<Pitch e' > #<Pitch gis'>)
 #<procedure line-markup (layout props args)>"+"))
((#<Pitch ees'> #<Pitch ges'>)
 #<procedure line-markup (layout props args)>"+")))
((#<Pitch ees'> #<Pitch ges'> #<Pitch bes'>)
 #<procedure line-markup (layout props args)>"ø")))
((#<Pitch ees'> #<Pitch ges'> #<Pitch beses'>)
 #<procedure concat-markup (layout props args)>"7")))
((#<Pitch e' > #<Pitch g' > #<Pitch b'> #<Pitch fis'' >)
 #<procedure line-markup (layout props args)>"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)>"7")))
((#<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)>"7")))
Chapter 2: Translation

"alt"))
((#<Pitch g'>)
  #<procedure line-markup (layout props args)>
  ((#<procedure super-markup (layout props arg)>
    "5"))
((#<Pitch g'> #<Pitch c' >)
  #<procedure line-markup (layout props args)>
  ((#<procedure super-markup (layout props arg)>
    "5"))
)
• Set context property chordNameFunction to ignatzek-chord-names.
• Set context property chordNameLowercaseMinor to #f.
• Set context property chordNameSeparator to:
  '('#<procedure hspace-markup (layout props amount)> 0.5)
• Set context property chordNoteNamer to '().
• Set context property chordPrefixSpacer to 0.
• Set context property chordRootNamer to note-name->markup.
• Set context property clefGlyph to "clefs.G".
• Set context property clefPosition to -2.
• Set context property clefTranspositionFormatter to clef-transposition-markup.
• Set context property codaMarkFormatter to #<procedure 5575a20b1a00 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4 (number context)>.
• Set context property completionFactor to unity-if-multimeasure.
• Set context property crescendoSpanner to 'hairpin.
• Set context property cueClefTranspositionFormatter to clef-transposition-markup.
• Set context property dalSegnoTextFormatter to format-dal-segno-text.
• Set context property decrescendoSpanner to 'hairpin.
• Set context property doubleRepeatBarType to "...:"
• Set context property doubleRepeatSegnoBarType to ".|.S.|:"
• Set context property drumStyleTable to #<hash-table 5575a20b1a00 29/61>.
• Set context property endRepeatBarType to ":|.
• Set context property endRepeatSegnoBarType to ":|.S"
• Set context property explicitClefVisibility to:
  #(\#t \#t \#t)
• Set context property explicitCueClefVisibility to:
  #(\#f \#t \#t)
• Set context property explicitKeySignatureVisibility to:
  #(\#t \#t \#t)
• Set context property extendersOverRests to \#t.
• Set context property extraNatural to \#t.
• Set context property figuredBassAlterationDirection to -1.
• Set context property figuredBassFormatter to format-bass-figure.
• Set context property figuredBassLargeNumberAlignment to 0.
• Set context property figuredBassPlusDirection to -1.
• Set context property figuredBassPlusStrokedAlist to:
  '((2 . "figbass.twoplus")
   (4 . "figbass.fourplus")
   (5 . "figbass.fiveplus")
   (6 . "figbass.sixstroked")
   (7 . "figbass.sevenstroked")
   (9 . "figbass.ninestroked"))

• Set context property fineBarType to "|.".
• Set context property fineSegnoBarType to "|.S".
• Set context property fineStartRepeatSegnoBarType to "|.S.|:".
• Set context property fineText to "Fine".
• Set context property fingeringOrientations to:
  '((up down)

• Set context property firstClef to #t.
• Set context property forbidBreakBetweenBarLines to #t.

• Set context property graceSettings to:
  '((Voice Stem direction 1)
   (Voice Slur direction -1)
   (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 context property harmonicAccidentals to #t.

• Set context property highStringOne to #t.

• Set context property initialTimeSignatureVisibility to:
  #(#f #t #t)

• Set context property instrumentTransposition to #<Pitch c'>.

• Set context property keepAliveInterfaces to:
  '((bass-figure-interface
    chord-name-interface
    cluster-beacon-interface
dynamic-interface
    fret-diagram-interface
    lyric-syllable-interface
    note-head-interface
tab-note-head-interface
    lyric-interface
    percent-repeat-interface
• Set context 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 context property lyricMelismaAlignment to -1.

• Set context property majorSevenSymbol to:
  `('#<procedure line-markup (layout props args)>
      ((#<procedure fontsize-markup (layout props increment arg)>
        3
        (#<procedure triangle-markup (layout props filled)>
          #f))))`

• Set context property measureBarType to "|".

• Set context property melismaBusyProperties to:
  `'(melismaBusy
      slurMelismaBusy
      tieMelismaBusy
      beamMelismaBusy
      completionBusy)`

• Set context property metronomeMarkFormatter to format-metronome-markup.

• Set context property middleCClefPosition to -6.

• Set context property middleCPosition to -6.

• Set context property minorChordModifier to:
  `'#<procedure simple-markup (layout props str)>>`
• Set context property noChordSymbol to:
  '('#<procedure simple-markup (layout props str)>
  "N.C.")
• Set context property noteNameFunction to note-name-markup.
• Set context property noteNameSeparator to "/".
• Set context property noteToFretFunction to determine-frets.
• Set context property partCombineTextsOnNote to #t.
• Set context property pedalSostenutoStrings to:
  '("Sost. Ped." "*Sost. Ped." ")
• Set context property pedalSostenutoStyle to 'mixed.
• Set context property pedalSustainStrings to:
  '("Ped." "*Ped." ")
• Set context property pedalSustainStyle to 'text.
• Set context property pedalUnaCordaStrings to:
  '("una corda" "tre corde")
• Set context property pedalUnaCordaStyle to 'text.
• Set context property predefinedDiagramTable to #f.
• Set context property printAccidentalNames to #t.
• Set context property printKeyCancellation to #t.
• Set context property printOctaveNames to #f.
• Set context property printPartCombineTexts to #t.
• Set context property proportionalNotationDuration to #<Mom 1/4>.
• Set context property quotedCueEventTypes to:
  'note-event
  rest-event
  tie-event
  beam-event
  tuplet-span-event
  tremolo-event)
• Set context property quotedEventTypes to:
  '(StreamEvent)
• Set context property rehearsalMarkFormatter to #<procedure 5575a20b18c0 at
 /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4
(number context)>
• Set context property rehearsalMark to 1.
• Set context property repeatCountVisibility to all-repeat-counts-visible.
• Set context property restNumberThreshold to 1.
• Set context 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 . 0.4)
(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))

(haydnturn
 (script-stencil feta "haydnturn" . "haydnturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (direction . 1))

(henzelongfermata
 (script-stencil feta "dhenzelongfermata"
 . "uhenzelongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (script-priority . 4000)
 (direction . 1))

(henzeshortfermata
 (script-stencil feta "dhenzeshortfermata"
 . "uhenzeshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (script-priority . 4000)
 (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 . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(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))
(outsidecomma
(avoid-slur . around)
(direction . 1)
(padding . 0.2)
(script-stencil feta "lcomma" . "rcomma"))
(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))
(semicolon
(script-stencil
 feta
 "dsemicolon"
"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 \ 0.4)
(avoid-slur \ around)
(script-priority \ 4000)
(direction \ 1))

(signumcongruentiae
(script-stencil
 feta
 "dsignumcongruentiae"
 .
 "usignumcongruentiae")
(padding \ 0.2)
(avoid-slur \ outside)
(direction \ 1))

(slashturn
(script-stencil feta "slashturn" \ "slashturn")
(padding \ 0.2)
(avoid-slur \ inside)
(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)
(script-priority . -50)
(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))
(uprall
(script-stencil feta "uprall" . "uprall")
(padding . 0.2)
(avoid-slur . around)
(direction . 1))
(varcoda
(script-stencil feta "varcoda" . "varcoda")
(padding . 0.2)
(avoid-slur . outside)
Chapter 2: Translation

• Set context property `sectionBarType` to "||".
• Set context property `segnoBarType` to "S".
• Set context property `segnoMarkFormatter` to `format-segno-mark-considering-bar-lines`.
• Set context property `segnoStyle` to 'mark.'
• Set context property `slashChordSeparator` to:
  '
'(#{<procedure simple-markup (layout props str)>
"/"})
• Set context property `soloIIIText` to "Solo II".
• Set context property `soloIText` to "Solo".
• Set context property `startRepeatBarType` to ".|:".
• Set context property `startRepeatSegnoBarType` to "S.|:".
• Set context property `stringNumberOrientations` to:
  '(up down)
• Set context property `stringOneTopmost` to #t.
• Set context property `stringTunings` to:
  '('#<Pitch e '>
  #<Pitch b>
  #<Pitch g>
  #<Pitch d>
  #<Pitch a,>
  #<Pitch e,>)

(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 . 0.4)
  (avoid-slur . around)
  (script-priority . 4000)
  (direction . 1))
(veryshortfermata
  (script-stencil
    feta
    "dveryshortfermata"
    .
    "uveryshortfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (script-priority . 4000)
  (direction . 1)))
• Set context property `strokeFingerOrientations` to: 
  'right'
• Set context property `subdivideBeams` to `#f`.
• Set context property `suspendMelodyDecisions` to `#f`.
• Set context property `systemStartDelimiter` to `SystemStartBar`.
• Set context property `tablatureFormat` to `fret-number-tablature-format`.
• Set context property `tabStaffLineLayoutFunction` to `tablature-position-on-lines`.
• Set context property `tieWaitForNote` to `#f`.
• Set context property `timeSignatureFraction` to: 
  '(4 . 4)
• Set context property `timeSignatureSettings` to:
  '(((2 . 2) (beamExceptions (end (1/32 8 8 8 8)))))
  ((3 . 2)
   (beamExceptions (end (1/32 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)
   (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)))))
  ((9 . 4)
   (beamExceptions (end (1/32 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 context property `timing` to `#t`.
• Set context property `topLevelAlignment` to `#t`.
• Set context property `underlyingRepeatBarType` to "||".

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Staff (page 287).

Context ChordGridScore can contain ChoirStaff (page 66), ChordGrid (page 68), ChordNames (page 95), Devnull (page 108), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

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

  Beam_collision_engraver (page 407)
  Help beams avoid colliding with notes and clefs in other voices.
Break_align_engraver (page 410)
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): BreakAlignGroup (page 500), BreakAlignment (page 501), and LeftEdge (page 571).

Centered_bar_number_align_engraver (page 411)
Group measure-centered bar numbers in a CenteredBarNumberLineSpanner so they end up on the same vertical position.
Properties (read)

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

This engraver creates the following layout object(s):
CenteredBarNumberLineSpanner (page 507).

Concurrent_hairpin_engraver (page 415)
Collect concurrent hairpins.

Footnote_engraver (page 423)
Create footnote texts.
This engraver creates the following layout object(s): Footnote (page 548).

Grace_spacing_engraver (page 425)
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): GraceSpacing (page 553).

Jump_engraver (page 428)
This engraver creates instructions such as D.C. and Fine, placing them vertically outside the set of staves given in the stavesFound context property.
If Jump_engraver is added or moved to another context, Staff_collecting_engraver (page 448), also needs to be there so that marks appear at the intended Y location.
Music types accepted: ad-hoc-jump-event (page 48), dal-segno-event (page 50), and fine-event (page 51),
Properties (read)

- codaMarkCount (non-negative, exact integer)
  Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

- codaMarkFormatter (procedure)
  A procedure that creates a coda mark (which in conventional D.S. al Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.
dalSegnoTextFormatter (procedure)
Format a jump instruction such as D.S.
The first argument is the context.
The second argument is the number of times the instruction is performed.
The third argument is a list of three markups: start-markup, end-markup, and next-markup.
If start-markup is #f, the form is da capo; otherwise the form is dal segno and start-markup is the sign at the start of the repeated section.
If end-markup is not #f, it is either the sign at the end of the main body of the repeat, or it is a Fine instruction. When it is a Fine instruction, next-markup is #f.
If next-markup is not #f, it is the mark to be jumped to after performing the body of the repeat, e.g., Coda.

finalFineTextVisibility (boolean)
Whether \fine at the written end of the music should create a Fine instruction.
fineText (markup)
The text to print at \fine.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.

stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): JumpScript (page 561).

Mark_engraver (page 432)
This engraver creates rehearsal marks, segno and coda marks, and section labels. Mark_engraver creates marks, formats them, and places them vertically outside the set of staves given in the stavesFound context property.
If Mark_engraver is added or moved to another context, Staff_collecting_engraver (page 448), also needs to be there so that marks appear at the intended Y location.
By default, Mark_engravers in multiple contexts create a common sequence of marks chosen by the Score-level Mark_tracking_translator (page 433). If independent sequences are desired, multiple Mark_tracking_translators must be used.

Properties (read)
codaMarkFormatter (procedure)
A procedure that creates a coda mark (which in conventional D.S. al Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.
currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

rehearsalMarkFormatter (procedure)
A procedure taking as arguments the context and the sequence number of the rehearsal mark. It should return the formatted mark as a markup object.

segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.

stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): CodaMark (page 515), RehearsalMark (page 607), SectionLabel (page 615), and SegnoMark (page 616).

Mark_tracking_translator (page 433)
This translator chooses which marks Mark_engraver should engrave.

Music types accepted: ad-hoc-mark-event (page 48), coda-mark-event (page 50), rehearsal-mark-event (page 55), section-label-event (page 55), and segno-mark-event (page 56).

Properties (read)
codaMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

rehearsalMark (integer)
The next rehearsal mark to print.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

Properties (write)
codaMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.
rehearsalMark (integer)
    The next rehearsal mark to print.

segnoMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a segno appears: not set
during the first timestep, 0 up to the first segno, 1 from the first to the
second segno, 2 from the second to the third segno, etc.

Metronome_mark_engraver (page 435)
    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 2.2.135
[Staff_collecting_engraver], page 448.
    Music types accepted: tempo-change-event (page 58),
Properties (read)
    currentCommandColumn (graphical (layout) object)
        Grob that is X-parent to all current breakable items (clef, key signature,
etc.).
    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): MetronomeMark (page 585).

Output_property_engraver (page 439)
    Apply a procedure to any grob acknowledged.
    Music types accepted: apply-output-event (page 48),

Paper_column_engraver (page 440)
    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: break-event (page 49), and label-event (page 52),
Properties (read)
    forbidBreak (boolean)
        If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.

Properties (write)
    currentCommandColumn (graphical (layout) object)
        Grob that is X-parent to all current breakable items (clef, key signature,
etc.).
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, except if explicitly re-quested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): NonMusicalPaperColumn (page 593), and PaperColumn (page 600).

Parenthesis_engraver (page 440)
Parenthesize objects whose parenthesize property is #t.
This engraver creates the following layout object(s): Parentheses (page 601).

Repeat_acknowledge_engraver (page 443)
This translator adds entries to repeatCommands for events generated by \repeat volta.

Music types accepted: volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59),

Properties (write)

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

Show_control_points_engraver (page 446)
Create grobs to visualize control points of Bézier curves (ties and slurs) for ease of tweaking.
This engraver creates the following layout object(s): ControlPoint (page 519), and ControlPolygon (page 520).

Spacing_engraver (page 447)
Make a SpacingSpanner and do bookkeeping of shortest starting and playing notes.

Music types accepted: spacing-section-event (page 56),

Properties (read)

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

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): SpacingSpanner (page 626).

Spanner_tracking_engraver (page 448)
Helper for creating spanners attached to other spanners. If a spanner has the sticky-grob-interface, the engraver tracks the spanner contained in its sticky-host object. When the host ends, the sticky spanner attached to it has its end announced too.

Staff_collecting_engraver (page 448)
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.

Stanza_number_align_engraver (page 449)
This engraver ensures that stanza numbers are neatly aligned.

Timing_translator (page 454)
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.
Music types accepted: alternative-event (page 48), bar-event (page 49), and fine-event (page 51),
Properties (read)

alternativeNumberingStyle (symbol)
The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

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 time-keeping, 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)

alternativeNumber (non-negative, exact integer)
When set, the index of the current alternative element, starting from
one. Not set outside of alternatives. Note the distinction from volta
number: an alternative may pertain to multiple volte.

baseMoment (moment)
Smallest unit of time that will stand on its own as a subdivided section.

currentBarNumber (integer)
Contains the current bar number. This property is incremented at every
bar line.

internalBarNumber (integer)
Contains the current bar number. This property is used for internal time-
keeping, 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 manu-
ally to create incomplete measures.

measureStartNow (boolean)
True at the beginning of a measure.

timeSignatureFraction (fraction, as pair)
A pair of numbers, signifying the time signature. For example, '(4 . 4)
is a 4/4 time signature.

Tweak_engraver (page 456)
Read the tweaks property from the originating event, and set properties.

Vertical_align_engraver (page 456)
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): StaffGrouper (page 630), and
VerticalAlignment (page 668).

Volta_engraver (page 456)
Make volta brackets.

Music types accepted: dal-segno-event (page 50), fine-event (page 51), and
volta-span-event (page 59),
Properties (read)

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

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '((start-repeat)).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

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): VoltaBracket (page 671), and VoltaBracketSpanner (page 672).

2.1.4 ChordNames
Typesets chord names.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): ChordName (page 508), StaffSpacing (page 632), and VerticalAxisGroup (page 668).

This context sets the following properties:
• Set grob property font-size in Parentheses (page 601), to 1.5.
• Set grob property nonstaff-nonstaff-spacing.padding in VerticalAxisGroup (page 668), to 0.5.
• Set grob property nonstaff-relatedstaff-spacing.padding in VerticalAxisGroup (page 668), to 0.5.
• Set grob property remove-empty in VerticalAxisGroup (page 668), to #t.
• Set grob property remove-first in VerticalAxisGroup (page 668), to #t.
• Set grob property staff-affinity in VerticalAxisGroup (page 668), 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):
Alteration_glyph_engraver (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.
Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group engraver (page 403)
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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Chord_name engraver (page 412)
Read currentChordText to create chord names.

Properties (read)

chordChanges (boolean)
Only show changes in chords scheme?

currentChordCause (stream event)
Event cause of the chord that should be created in this time step (if any).

currentChordText (markup)
In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

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

Properties (write)

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

This engraver creates the following layout object(s): ChordName (page 508).

Current_chord_text engraver (page 416)
Catch note and rest events and generate the appropriate chord text using chordNameFunction. Actually creating a chord name grob is left to other engravers.

Music types accepted: general-rest-event (page 51), and note-event (page 54),

Properties (read)

chordNameExceptions (list)
An list 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.

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)

currentChordCause (stream event)
Event cause of the chord that should be created in this time step (if any).

currentChordText (markup)
In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Separating_line_group_engraver (page 445)
Generate objects for computing spacing parameters.

Properties (read)

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

Properties (write)

hasStaffSpacing (boolean)
True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

2.1.5 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 (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Fingering (page 545), Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570),
This context sets the following properties:

- Set context property `fontSize` to -4.
- Set grob property `beam-thickness` in `Beam` (page 495), to 0.35.
- Set grob property `beam-thickness` in `StemTremolo` (page 637), to 0.35.
- Set grob property `ignore-ambitus` in `NoteHead` (page 596), to #t.
- Set grob property `length-fraction` in `Beam` (page 495), to 0.6299605249474366.
- Set grob property `length-fraction` in `Stem` (page 634), to 0.6299605249474366.

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):

- **Arpeggio_engraver** (page 402)
  
  Generate an Arpeggio symbol.

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

- **Auto_beam_engraver** (page 402)

  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 2.2.141 [Stem_engraver], page 450, properties `stemLeftBeamCount` and `stemRightBeamCount`.

  *Music types accepted: beam-forbid-event (page 49),*

  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 `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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams.
Music types accepted: beam-event (page 49),
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): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.
Music types accepted: bend-after-event (page 49),
Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

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

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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)

breathMarkType (symbol)
The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).
Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514),
and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-
interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
quested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat
(page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner
(page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-
event (page 49), and span-dynamic-event (page 56),
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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.
Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

Fingering_engraver (page 422)
Create fingering scripts.
Music types accepted: fingering-event (page 51),
This engraver creates the following layout object(s): Fingering (page 545).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Forbid_line_break_engraver (page 423)
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, except if explicitly requested by the user.

Glissando_engraver (page 424)
Engrave glissandi.
Music types accepted: glissando-event (page 51),
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): Glissando (page 552).

Grace_auto_beam_engraver (page 424)
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: beam-forbid-event (page 49),
Properties (read)

  autoBeaming (boolean)
  If set to true then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).

Grace_beam_engraver (page 425)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engravess beams when we are at grace points in time.
Music types accepted: beam-event (page 49),
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): Beam (page 495).

Grace_engraver (page 425)
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.

Grob_pq_engraver (page 426)
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.).
**Instrument_switch_engraver** (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

- **instrumentCueName** (markup)
  The name to print if another instrument is to be taken.
  This property is deprecated

This engraver creates the following layout object(s): **InstrumentSwitch** (page 560).

**Laissez_vibrer_engraver** (page 430)
Create laissez vibrer items.
Music types accepted: **laissez-vibrer-event** (page 52),
This engraver creates the following layout object(s): **LaissezVibrerTie** (page 569), and **LaissezVibrerTieColumn** (page 570).

**Ligature_bracket_engraver** (page 431)
Handle Ligature_events by engraving Ligature brackets.
Music types accepted: **ligature-event** (page 52),
This engraver creates the following layout object(s): **LigatureBracket** (page 573).

**Multi_measure_rest_engraver** (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.
Music types accepted: **multi-measure-articulation-event** (page 53), **multi-measure-rest-event** (page 53), and **multi-measure-text-event** (page 53),
Properties (read)

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

- **internalBarNumber** (integer)
  Contains the current bar number. This property is used for internal time-keeping, among others by the **Accidental_engraver**.

- **measureStartNow** (boolean)
  True at the beginning of a measure.

- **restNumberThreshold** (number)
  If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): **MultiMeasureRest** (page 587), **MultiMeasureRestNumber** (page 589), **MultiMeasureRestScript** (page 590), and **MultiMeasureRestText** (page 592).

**New_fingering_engraver** (page 437)
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.
Chapter 2: Translation

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): Fingering (page 545), Script (page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)
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): VoiceFollower (page 670).

Note_heads_engraver (page 438)
Generate note heads.

Music types accepted: note-event (page 54),

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): NoteHead (page 596).

Note_spacing_engraver (page 439)
Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: note-event (page 54), and part-combine-event (page 54),

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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)
  countPercentRepeats (boolean)
    If set, produce counters for percent repeats.
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
Print the bracketed note head after a note head with trill.
This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).
Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will
fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash
(page 535), and RepeatSlash (page 609).

Slur_engraver (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Stem_engraver (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam
engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event
(page 58),
Properties (read)
   currentBarLine (graphical (layout) object)
      Set to the BarLine that Bar_engraver has created in the current
timestep.
   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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

Text_ engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_ engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).

Tie_ engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and TieColumn (page 655).

Trill_spanner_ engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)

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

  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): TrillSpanner (page 661).
Tuplet_engraver (page 455)
Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 58),

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): TupletBracket (page 662), and TupletNumber (page 664).

2.1.6 Devnull
Silently discards all musical information given to this context.

This context also accepts commands for the following context(s): Staff (page 287), and Voice (page 390).

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.

2.1.7 DrumStaff
Handles typesetting for percussion.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), BreathingSign (page 502), CaesuraScript (page 504), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), LedgerLineSpanner (page 570), NoteCollision (page 595), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedalLineSpanner (page 642), TimeSignature (page 655), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:
- Set context property `clefGlyph` to "clefs.percussion".
- Set context property `clefPosition` to 0.
- Set context property `createSpacing` to #t.
- Set context property `ignoreFiguredBassRest` to #f.
- Set context property `instrumentName` to ". ()".
- Set context property `localAlterations` to ". ()".
- Set context property `ottavationMarkups` to:
  `'(4 , "29")`
  `(3 , "22")`
  `(2 , "15")`
  `(1 , "8")`
• Set context property shortInstrumentName to 

• Set grob property staff-padding in Script (page 613), to 0.75.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type DrumVoice (page 117).

Context DrumStaff can contain CueVoice (page 97), DrumVoice (page 117), and NullVoice (page 228).

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

Alteration_glyph_engraver (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.

Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)
Create bar lines for various commands, including \\bar.
If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.
Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

caesuraType (list)
An alist

((bar-line . bar-type)
specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:...’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.S.|’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.S’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.|.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.|S.|’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.
repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '(start-repeat) may be given as '(start-repeate).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
The current bar line type, or ’() if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

\currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

\forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
Notate a short break in sound that does not shorten the previous note.
Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.

If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),

Properties (read)

- **breathMarkDefinitions** (list)
  The description of breath marks. This is used by the Breathing_sign_ engraver. See scm/breath.scm for more information.

- **caesuraType** (list)
  An alist
  
  ```lisp
  ((bar-line . bar-type)
   (breath . breath-type)
   (scripts . script-type...)
   (underlying-bar-line . bar-type))
  ```

  specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

  bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

- **caesuraTypeTransform** (procedure)
  An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

  The first argument is the context.

  The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

  The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

- **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): BreathingSign (page 502), and CaesuraScript (page 504).

**Clef_engraver** (page 413)
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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.
forceBreak (boolean)
   Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513),
CueClef (page 521), and CueEndClef (page 524).

Dot_column_engraver (page 418)
   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): DotColumn (page 531).

Figured_bass_engraver (page 421)
   Make figured bass numbers.
   Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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 exten-
der lines.
   useBassFigureExtenders (boolean)
      Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
   Position figured bass alignments over notes.
   This engraver creates the following layout object(s):
   BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
   Find potentially colliding scripts and put them into a FingeringColumn object; that
will fix the collisions.
   This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
   Put fontSize into font-size grob property.
Properties (read)
   fontSize (number)
      The relative size of all grobs in a context.
Grob_pq_engraver (page 426)
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.).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

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): InstrumentName (page 559).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines.
This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),
Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.
Properties (read)

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

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
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): RestCollision (page 612).

Script_row_engraver (page 445)
Determine order in horizontal side position elements.
This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
Generate objects for computing spacing parameters.
Properties (read)

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

Properties (write)

- hasStaffSpacing (boolean)
  True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
Create a StaffEllipsis when skipTypesetting is used.
Properties (read)

- skipTypesetting (boolean)
  If true, no typesetting is done, speeding up the interpretation phase.
  Useful for debugging large scores.

This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),

Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),
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): TimeSignature (page 655).

2.1.8 DrumVoice

A voice on a percussion staff.

This context also accepts commands for the following context(s): Voice (page 390).

This context creates the following layout object(s): Beam (page 495), BendAfter (page 497), BreathingSign (page 502), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Flag (page 547), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), Stem (page 634), StemStub (page 636), StemTremolo (page 637), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655),
TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), and TupletNumber (page 664).

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):

**Auto_beam_engraver** (page 402)

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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 49),

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 alst 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 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): Beam (page 495).

**Beam_engraver** (page 407)

Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted: beam-event (page 49),

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): Beam (page 495).
Bend_engraver (page 409)
Create fall spanners.
Music types accepted: bend-after-event (page 49),
Properties (read)
  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
  currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)
  breathMarkType (symbol)
  The type of BreathingSign to create at \breathe.
This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly requested by the user.
This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

Drum_notes_engraver (page 419)
Generate drum note heads.

Music types accepted: note-event (page 54),

Properties (read)

\[
\text{drumStyleTable (hash table)}
\]

A hash table which maps drums to layout settings. Predefined values:

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): NoteHead (page 596), and Script (page 613).

Dynamic_align_engraver (page 419)
Align hairpins and dynamic texts on a horizontal line.

Properties (read)

\[
\text{currentMusicalColumn (graphical (layout) object)}
\]

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

This engraver creates the following layout object(s): DynamicLineSpanner (page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56),

Properties (read)

\[
\text{crescendoSpanner (symbol)}
\]

The type of spanner to be used for crescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin crescendo is used.

\[
\text{crescendoText (markup)}
\]

The text to print at start of non-hairpin crescendo, i.e., ‘cresc.’.

\[
\text{currentMusicalColumn (graphical (layout) object)}
\]

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

\[
\text{decrescendoSpanner (symbol)}
\]

The type of spanner to be used for decrescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin decrescendo is used.

\[
\text{decrescendoText (markup)}
\]

The text to print at start of non-hairpin decrescendo, i.e., ‘dim.’.

This engraver creates the following layout object(s): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).
Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.
Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner
(page 543).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)
fontSize (number)
The relative size of all grobs in a context.

Forbid_line_break_engraver (page 423)
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, except if explicitly re-
quested by the user.

Grace_auto_beam_engraver (page 424)
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: beam-forbid-event (page 49),
Properties (read)
autoBeaming (boolean)
If set to true then beams are generated automatically.
This engraver creates the following layout object(s): Beam (page 495).

Grace_beam_engraver (page 425)
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: beam-event (page 49),
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): Beam (page 495).

**Grace_engraver (page 425)**

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.

**Grob_pq_engraver (page 426)**

Administrate when certain grobs (e.g., note heads) stop playing.

**Properties (read)**

busyGros (list)

A queue of \((\text{end-moment} \cdot \text{grob})\) 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 \((\text{end-moment} \cdot \text{grob})\) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g., note heads, spanners, etc.).

**Instrument_switch_engraver (page 427)**

Create a cue text for taking instrument.

This engraver is deprecated.

**Properties (read)**

instrumentCueName (markup)

The name to print if another instrument is to be taken.

This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

**Laissez_vibrer_engraver (page 430)**

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 52),

This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

**Multi_measure_rest_engraver (page 436)**

Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.

Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),

**Properties (read)**

currentCommandColumn (graphical (layout) object)

Grob that is X-parent to all current breakable items (clef, key signature, etc.).

internalBarNumber (integer)

Contains the current barnumber. This property is used for internal time-keeping, among others by the Accidental_engraver.
measureStartNow (boolean)
  True at the beginning of a measure.

restNumberThreshold (number)
  If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587),
MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and
MultiMeasureRestText (page 592).

Note_spacing_engraver (page 439)
  Generate NoteSpacing, an object linking horizontal lines for use in spacing.
  This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
  Apply a procedure to any grob acknowledged.
  Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
  Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and
  ‘unisono’.
  Music types accepted: note-event (page 54), and part-combine-event (page 54),
  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): CombineTextScript
(page 517).

Percent_repeat_engraver (page 441)
  Make whole measure repeats.
  Music types accepted: percent-event (page 54),
  Properties (read)
    countPercentRepeats (boolean)
      If set, produce counters for percent repeats.
    currentCommandColumn (graphical (layout) object)
      Grob that is X-parent to all current breakable items (clef, key signature,
      etc.).
    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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
Print the bracketed note head after a note head with trill.
This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).

Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

**Slur_engraver** (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

**Spanner_break_forbid_engraver** (page 448)
Forbid breaks in certain spanners.

**Stem_engraver** (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)
- **currentBarLine** (graphical (layout) object)
  - Set to the BarLine that Bar_engraver has created in the current timestep.
- **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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

**Text_engraver** (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

**Text_spanner_engraver** (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).
Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and
TieColumn (page 655).

Trill_spanner_engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature,
    etc.).
  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): TrillSpanner (page 661).

Tuplet_engraver (page 455)
Catch tuplet events and generate appropriate bracket.
Music types accepted: tuplet-span-event (page 58),
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 signa-
    tures, etc.) before the note.
This engraver creates the following layout object(s): TupletBracket (page 662),
and TupletNumber (page 664).

2.1.9 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): Staff (page 287), and
Voice (page 390).
  This context creates the following layout object(s): BarLine (page 486),
DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541),
Hairpin (page 555), PianoPedalBracket (page 606), Script (page 613), SostenutoPedal (page 623), SustainPedal (page 641), TextScript (page 650), TextSpanner (page 652), UnaCordaPedal (page 665), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set context property pedalSustainStrings to:
  `'("Ped." "+Ped." "+")`
- Set context property pedalUnaCordaStrings to:
  `'("una corda" "tre corde")`
- Set grob property font-shape in TextScript (page 650), to ‘italic.
- Set grob property nonstaff-relatedstaff-spacing in VerticalAxisGroup (page 668), to:
  `'((basic-distance . 5) (padding . 0.5))`
- Set grob property outside-staff-priority in DynamicLineSpanner (page 538), to #f.
- Set grob property outside-staff-priority in DynamicText (page 539), to #f.
- Set grob property outside-staff-priority in Hairpin (page 555), to #f.
- Set grob property staff-affinity in VerticalAxisGroup (page 668), to 0.
- Set grob property Y-offset in DynamicLineSpanner (page 538), 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):

Axis_group_engraver (page 403)

- 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 items (clef, key signature, etc.).

- 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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)

- Create bar lines for various commands, including `\\bar`.

  If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

  Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59),
Properties (read)

caesuraType (list)
An alist

\((\text{bar-line . bar-type})\)
\((\text{breath . breath-type})\)
\((\text{scripts . script-type...})\)
\((\text{underlying-bar-line . bar-type})\)

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

\text{bar-line} has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. \text{bar-line} indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:..’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.S.|:’.

derendRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

derendRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.S’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|..S’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|..S.|:’.
forbidBreakBetweenBarLines (boolean)
   If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
   Bar line to insert at a measure boundary.

repeatCommands (list)
   A list of commands related to volta-style repeats. In general, each element is a list, '
   (command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '
   ((start-repeat)) may be given as '(start-repeat).
   end-repeat
      End a repeated section.
   start-repeat
      Start a repeated section.
   volta text
      If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
   Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '||'.

segnoBarType (string)
   Bar line to insert at an in-staff segno. The default is 'S'.

segnoStyle (symbol)
   A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
   Bar line to insert at the start of a \repeat volta. The default is ' .:'.

startRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is 'S .:'.

underlyingRepeatBarType (string)
   Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '||'.

whichBar (string)
   The current bar line type, or '( ) if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

currentBarLine (graphical (layout) object)
   Set to the BarLine that Bar_engraver has created in the current timestep.
forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner (page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56),

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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.

Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),
Properties (read)

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

`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): `PianoPedalBracket` (page 606), `SostenutoPedal` (page 623), `SustainPedal` (page 641), and `UnaCordaPedal` (page 665).

`Script_engraver` (page 445)
Handle note scripted articulations.

Music types accepted: `articulation-event` (page 49),

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): `Script` (page 613).

`Text_engraver` (page 452)
Create text scripts.

Music types accepted: `text-script-event` (page 58),

This engraver creates the following layout object(s): `TextScript` (page 650).

`Text_spanner_engraver` (page 452)
Create text spanner from an event.

Music types accepted: `text-span-event` (page 58),

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): `TextSpanner` (page 652).
2.1.10 FiguredBass

A context for printing a figured bass line.

This context creates the following layout object(s): BassFigure (page 491), BassFigureAlignment (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), StaffSpacing (page 632), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set grob property nonstaff-nonstaff-spacing.padding in VerticalAxisGroup (page 668), to 0.5.
- Set grob property nonstaff-relatedstaff-spacing.padding in VerticalAxisGroup (page 668), to 0.5.
- Set grob property remove-empty in VerticalAxisGroup (page 668), to #t.
- Set grob property remove-first in VerticalAxisGroup (page 668), to #t.
- Set grob property staff-affinity in VerticalAxisGroup (page 668), 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):

Axis_group_engraver (page 403)

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 items (clef, key signature, etc.).
- 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): VerticalAxisGroup (page 668).

Figured_bass_engraver (page 421)

Make figured bass numbers.

Music types accepted: bass-figure-event (page 49), and rest-event (page 55).

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): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Separating_line_group_engraver (page 445)
   Generate objects for computing spacing parameters.

Properties (read)
   createSpacing (boolean)
      Create StaffSpacing objects? Should be set for staves.

Properties (write)
   hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

2.1.11 FretBoards
   A context for displaying fret diagrams.

   This context also accepts commands for the following context(s): Staff (page 287).

   This context creates the following layout object(s): FretBoard (page 550),
   InstrumentName (page 559), StaffSpacing (page 632), and VerticalAxisGroup (page 668).

   This context sets the following properties:
   • Set context property handleNegativeFrets to 'recalculate.'
   • Set context property instrumentName to '().
   • Set context property predefinedDiagramTable to #<hash-table 5575a316c480 0/113>.
   • Set context property restrainOpenStrings to #f.
   • Set context 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):

   Axis_group_engraver (page 403)
      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 items (clef, key signature,
         etc.).

      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): VerticalAxisGroup (page 668).

Font_size engraver (page 422)
Put fontSize into font-size grob property.

Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Fretboard engraver (page 423)
Generate fret diagram from one or more events of type NoteEvent.
Music types accepted: fingering-event (page 51), note-event (page 54), and string-number-event (page 57).

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): FretBoard (page 550).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

  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): InstrumentName (page 559).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Separating_line_group_engraver (page 445)
Generate objects for computing spacing parameters.
Properties (read)

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

Properties (write)

  hasStaffSpacing (boolean)
  True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

2.1.12 Global
Hard coded entry point for LilyPond. Usually not meant to be modified directly.

  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 Score (page 263).

  Context Global can contain ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.1.13 GrandStaff
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): Arpeggio (page 483), InstrumentName (page 559), SpanBar (page 626), SpanBarStub (page 628), StaffGrouper (page 630),
SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646),
SystemStartSquare (page 647), and VerticalAlignment (page 668).

This context sets the following properties:

- Set context property instrumentName to '().
- Set context property localAlterations to #f.
- Set context property localAlterations to '().
- Set context property localAlterations to '().
- Set context property shortInstrumentName to '().
- Set context property systemStartDelimiter to 'SystemStartBrace.
- Set context property systemStartDelimiter to 'SystemStartBracket.
- Set context property topLevelAlignment to #f.
- Set grob property extra-spacing-width in DynamicText (page 539), to #f.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit
context of type Staff (page 287).

Context GrandStaff can contain ChoirStaff (page 66), ChordNames (page 95),
Devnull (page 108), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132),
FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137),
GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199),
MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230),
PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287),
StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and
VaticanaStaff (page 366).

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

Instrument_name_engraver (page 427)

  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 items (clef, key signature,
    etc.).

  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): InstrumentName (page 559).

Output_property_engraver (page 439)

  Apply a procedure to any grob acknowledged.

  Music types accepted: apply-output-event (page 48),
Span_arpeggio_engraver (page 447)
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): Arpeggio (page 483).

Span_bar_engraver (page 448)
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): SpanBar (page 626).

Span_bar_stub_engraver (page 448)
Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s): SpanBarStub (page 628).

System_start_delimiter_engraver (page 450)
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 items (clef, key signature, etc.).

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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

Vertical_align_engraver (page 456)
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): StaffGrouper (page 630), and VerticalAlignment (page 668).

2.1.14 GregorianTranscriptionLyrics
A lyrics context for notating Gregorian chant in modern style.

This context also accepts commands for the following context(s): Lyrics (page 199).

This context creates the following layout object(s): InstrumentName (page 559), LyricExtender (page 574), LyricHyphen (page 575), LyricRepeatCount
This context sets the following properties:

- Set context property `instrumentName` to '().
- Set context property `lyricRepeatCountFormatter` to `<procedure 5575a3702120 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:210:4 (context repeat-count)>`.
- Set context property `searchForVoice` to #f.
- Set context property `shortInstrumentName` to '().
- Set grob property `bar-extent` in `BarLine` (page 486), to:
  `'(-0.05 . 0.05)`
- Set grob property `font-size` in `InstrumentName` (page 559), to 1.0.
- Set grob property `nonstaff-nonstaff-spacing` in `VerticalAxisGroup` (page 668), to:
  `'((basic-distance . 0)
   (minimum-distance . 2.8)
   (padding . 0.2)
   (stretchability . 0))`
- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 668), to:
  `'((basic-distance . 5.5)
   (padding . 0.5)
   (stretchability . 1))`
- Set grob property `nonstaff-unrelatedstaff-spacing.padding` in `VerticalAxisGroup` (page 668), to 1.5.
- Set grob property `parent-alignment-X` in `LyricRepeatCount` (page 576), to 1.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 668), to #t.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 668), to #t.
- Set grob property `self-alignment-Y` in `InstrumentName` (page 559), to #f.
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 668), 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):

`Axis_group_engraver` (page 403)

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 items (clef, key signature, etc.).

- `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): VerticalAxisGroup (page 668).

Extender_engraver (page 421)
Create lyric extenders.
Music types accepted: completize-extender-event (page 50), and extender-event (page 51),
Properties (read)

extendersOverRests (boolean)
Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): LyricExtender (page 574).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Hyphen_engraver (page 427)
Create lyric hyphens, vowel transitions and distance constraints between words.
Music types accepted: hyphen-event (page 52), and vowel-transition-event (page 59),
This engraver creates the following layout object(s): LyricHyphen (page 575), LyricSpace (page 578), and VowelTransition (page 673).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

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): InstrumentName (page 559).

Lyric_engraver (page 431)
Engrave text for lyrics.
Music types accepted: lyric-event (page 52),
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): LyricText (page 579).

Lyric_repeat_count_engraver (page 431)
Create repeat counts within lyrics for modern transcriptions of Gregorian chant.
Music types accepted: volta-repeat-end-event (page 59),
Properties (read)

lyricRepeatCountFormatter (procedure)
A procedure taking as arguments the context and the numeric repeat
count. It should return the formatted repeat count as markup. If it does
not return markup, no grob is created.

This engraver creates the following layout object(s): LyricRepeatCount (page 576).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Stanza_number_engraver (page 450)
Engrave stanza numbers.
Properties (read)

stanza (markup)
Stanza ‘number’ to print before the start of a verse. Use in Lyrics con-
text.

This engraver creates the following layout object(s): StanzaNumber (page 633).

2.1.15 GregorianTranscriptionStaff
A staff for notating Gregorian chant in modern style.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): Accidental (page 475),
AccidentalCautionary (page 476), AccidentalPlacement (page 477),
AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492),
BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine
(page 494), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef
(page 524), Divisio (page 528), DotColumn (page 531), FingeringColumn (page 547),
InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565),
LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599),
PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614),
ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624),
StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632),
StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642),
UnaCordaPedal (page 665), UnaCordaPedallLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set context property autoAccidentals to: '
(Staff #procedure 5575a36e3080 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0

- Set context property autoCautionaries to '().

- Set context property caesuraTypeTransform to caesura-to-bar-line-or-divisio.

- Set context property caesuraType to:
'((breath . varcomma))

- Set context property createSpacing to #t.

- Set context property doubleRepeatBarType to "||".

- Set context property doubleRepeatSegnoBarType to "S-||".

- Set context property endRepeatBarType to "||".

- Set context property endRepeatSegnoBarType to "S-||".

- Set context property extraNatural to #f.

- Set context property fineBarType to "||".

- Set context property fineSegnoBarType to "S-||".

- Set context property fineStartRepeatSegnoBarType to "S-||".

- Set context property forbidBreakBetweenBarLines to #f.

- Set context property ignoreFiguredBassRest to #f.

- Set context property instrumentName to '().

- Set context property localAlterations to '().

- Set context property measureBarType to '().

- Set context property ottavationMarkups to:
'((4 . "29")
 (3 . "22")
 (2 . "15")
 (1 . "8")
 (-1 . "8")
 (-2 . "15")
 (-3 . "22")
 (-4 . "29"))

- Set context property printKeyCancellation to #f.

- Set context property sectionBarType to "||".

- Set context property segnoBarType to "S-||".

- Set context property shortInstrumentName to '().

- Set context property startRepeatBarType to "||".

- Set context property startRepeatSegnoBarType to "S-||".

- Set context property underlyingRepeatBarType to "||".

- Set grob property extra-spacing-height in BreathingSign (page 502), to item::extra-spacing-height-including-staff.

- Set grob property extra-spacing-width in BreathingSign (page 502), to:
'(-1.0 . 0.0)
This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type GregorianTranscriptionVoice (page 153).

Context GregorianTranscriptionStaff can contain CueVoice (page 97), GregorianTranscriptionVoice (page 153), and NullVoice (page 228).

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

Accidental_engraver (page 400)

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.

tautoAccidentals (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.

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.

tautoCautionaries (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 time-
keeping, 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): Accidental (page 475),
AccidentalCautionary (page 476), AccidentalPlacement (page 477), and
AccidentalSuggestion (page 478).

Alteration_glyph_engraver (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface
to the value of the context’s alterationGlyphs property, when defined.

Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as
exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print
accidentals.

Axis_group_engraver (page 403)
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 items (clef, key signature,
etc.).

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): VerticalAxisGroup
(page 668).
Bar_engraver (page 404)
Create bar lines for various commands, including `\bar`.

If `forbidBreakBetweenBarLines` is true, allow line breaks at bar lines only.

Music types accepted: `ad-hoc-jump-event` (page 48), `caesura-event` (page 50),
`coda-mark-event` (page 50), `dal-segno-event` (page 50), `fine-event` (page 51),
`section-event` (page 55), `segno-mark-event` (page 56), and `volta-span-event` (page 59),

Properties (read)

`caesuraType` (list)
An alist

```
((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))
```

specifying which breath mark, bar line, and scripts to create at `\caesura`. All entries are optional.

`bar-line` has higher priority than a measure bar line and `underlying-bar-line` has lower priority than a measure bar line.

`caesuraTypeTransform` (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as `caesuraType`.

The first argument is the context.

The second argument is the value of `caesuraType` with an additional entry `(articulations . symbol-list)` identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. `bar-line` indicates that the engraver has observed a BarLine at the current moment.

`doubleRepeatBarType` (string)
Bar line to insert where the end of one `\repeat volta` coincides with the start of another. The default is `':..:'`. 

`doubleRepeatSegnoBarType` (string)
Bar line to insert where an in-staff segno coincides with the end of one `\repeat volta` and the beginning of another. The default is `':|.S.|:'`. 

`endRepeatBarType` (string)
Bar line to insert at the end of a `\repeat volta`. The default is `':.'`. 

`endRepeatSegnoBarType` (string)
Bar line to insert where an in-staff segno coincides with the end of a `\repeat volta`. The default is `':|.S.'`. 

`fineBarType` (string)
Bar line to insert at `\fine`. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is `'|.'`. 

fineSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with \fine. The default is ‘| .S’.

fineStartRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘| .S .| :’.

forbidBreakBetweenBarLines (boolean)
   If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
   Bar line to insert at a measure boundary.

repeatCommands (list)
   A list of commands related to volta-style repeats. In general, each element is a list, '(command args...)', but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat))' may be given as '(start-repeat).

   end-repeat
      End a repeated section.

   start-repeat
      Start a repeated section.

   volta text
      If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
   Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
   Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
   A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
   Bar line to insert at the start of a \repeat volta. The default is ‘.| ::’.

startRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S .| ::’.

underlyingRepeatBarType (string)
   Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
   The current bar line type, or ‘()’ if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.
Properties (write)

- **currentBarLine** (graphical (layout) object)
  - Set to the BarLine that Bar_engraver has created in the current timestep.

- **forbidBreak** (boolean)
  - If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

**Clef_engraver** (page 413)

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.

- **forbidBreak** (boolean)
  - If set to #t, prevent a line break at this point, except if explicitly requested by the user.

- **forceBreak** (boolean)
  - Set to #t when an event forcing a line break was heard.

- **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): Clef (page 510), and ClefModifier (page 513).

**Collision_engraver** (page 413)

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): NoteCollision (page 595).

**Cue_clef_engraver** (page 415)

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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

Divisio_engraver (page 417)
Create divisiones: chant notation for points of breathing or caesura.

Music types accepted: caesura-event (page 50), fine-event (page 51), section-event (page 55), volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59),

Properties (read)
caesuraType (list)
An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

The first argument is the context.

The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set;
the function is free to return a different value. The transform function
can remove articulations, but any added articulations are ignored. 
The third argument is a symbol-list identifying certain things the en-
graver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

This engraver creates the following layout object(s): Divisio (page 528).

Dot_column_engraver (page 418)
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): DotColumn (page 531).

Figured_bass_engraver (page 421)
Make figured bass numbers.
Music types accepted: bass-figure-event (page 49), and rest-event (page 55), 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 exten-
der lines.

useBassFigureExtenders (boolean)
Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.
This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that
will fix the collisions.
This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.
Grob\_pq\_engraver (page 426)
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

\texttt{busyGrobs (list)}
A queue of \texttt{(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)

\texttt{busyGrobs (list)}
A queue of \texttt{(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.).

Instrument\_name\_engraver (page 427)
Create a system start text for instrument or vocal names.

Properties (read)

\texttt{currentCommandColumn (graphical (layout) object)}
Grob that is X-parent to all current breakable items (clef, key signature, etc.).

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

\texttt{shortInstrumentName (markup)}
See \texttt{instrumentName}.

\texttt{shortVocalName (markup)}
Name of a vocal line, short version.

\texttt{vocalName (markup)}
Name of a vocal line.

This engraver creates the following layout object(s): \texttt{InstrumentName} (page 559).

Key\_engraver (page 429)
Engrave a key signature.

Music types accepted: \texttt{key-change-event} (page 52),

Properties (read)

\texttt{createKeyOnClefChange (boolean)}
Print a key signature whenever the clef is changed.

\texttt{explicitKeySignatureVisibility (vector)}
\texttt{\'break-visibility\'} function for explicit key changes. \texttt{\'\override\'} of the \texttt{break-visibility} property will set the visibility for normal (i.e., at the start of the line) key signatures.

\texttt{extraNatural (boolean)}
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

\texttt{forbidBreak (boolean)}
If set to \texttt{#t}, prevent a line break at this point, except if explicitly requested by the user.
forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

keyAlterationOrder (list)
A list of pairs that defines in what order alterations should be printed.
The format of an entry is (step . alter), where step is a number from
0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
rationals for alterations in between, e.g., 1/2 for sharp.

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): KeyCancellation (page 563),
and KeySignature (page 565).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines.
This engraver creates the following layout object(s): LedgerLineSpanner
(page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all
have the same text and there are at least two only the first one is retained and the
others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn
object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).
Ottava_spanner_engraver (page 439)
Create a text spanner when the ottavation property changes.
Music types accepted: ottava-event (page 54),
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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.
Properties (read)
currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),
Properties (read)
currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
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): PianoPedalBracket
(page 606), SostenutoPedal (page 623), SustainPedal (page 641), and
UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)
   Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
   Handle collisions of rests.

   Properties (read)
   busyGrobs (list)
      A queue of \texttt{(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): RestCollision (page 612).

Script_row_engraver (page 445)
   Determine order in horizontal side position elements.

   This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
   Generate objects for computing spacing parameters.

   Properties (read)
   createSpacing (boolean)
      Create StaffSpacing objects? Should be set for staves.

   Properties (write)
   hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.

   This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
   Create a StaffEllipsis when skipTypesetting is used.

   Properties (read)
   skipTypesetting (boolean)
      If true, no typesetting is done, speeding up the interpretation phase.
      Useful for debugging large scores.

   This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
   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_highlight_ engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),
Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_ engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

2.1.16 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 (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), Episema (page 542), FingerGlideSpanner (page 543), Fingering (page 545), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), LigatureBracket (page 573), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), and VoiceFollower (page 670).

This context sets the following properties:

- Set context property autoBeaming to #f.

- Set grob property transparent in LigatureBracket (page 573), to #t.

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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.
Music types accepted: arpeggio-event (page 49),
This engraver creates the following layout object(s): Arpeggio (page 483).
Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 49),

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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams.

Music types accepted: beam-event (page 49),

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): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.

Music types accepted: bend-after-event (page 49),

Properties (read)

  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
currentCommandColumn (graphical (layout) object)
   Grob that is X-parent to all current breakable items (clef, key signature, etc.).

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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
   Notate breath marks.
   Music types accepted: breathing-event (page 50),
   Properties (read)
      breathMarkType (symbol)
      The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
   Generate beams for tremolo repeats.
   Music types accepted: tremolo-span-event (page 58),
   This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_engraver (page 413)
   Engrave a cluster using Spanner notation.
   Music types accepted: cluster-note-event (page 50),
   This engraver creates the following layout object(s): ClusterSpanner (page 514),
   and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
   Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
   This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
   Make double measure repeats.
   Music types accepted: double-percent-event (page 50),
   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, except if explicitly requested by the user.
This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

**Dynamic_align_engraver** (page 419)
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): DynamicLineSpanner (page 538).

**Dynamic_engraver** (page 420)
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56).

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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

**Episema_engraver** (page 421)
Create an Editio Vaticana-style episema line.

Music types accepted: episema-event (page 51),

This engraver creates the following layout object(s): Episema (page 542).

**Finger_glide_engraver** (page 422)
Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),

This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

**Fingering_engraver** (page 422)
Create fingering scripts.

Music types accepted: fingering-event (page 51),

This engraver creates the following layout object(s): Fingering (page 545).
Font size engraver (page 422)
   Put fontSize into font-size grob property.
   Properties (read)
      fontSize (number)
      The relative size of all grobs in a context.

Forbid line break engraver (page 423)
   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, except if explicitly re-
      quested by the user.

Glissando engraver (page 424)
   Engrave glissandi.
   Music types accepted: glissando-event (page 51),
   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 nu-
      mber of note-heads in the two note columns between which the glissandi
      occur.

   This engraver creates the following layout object(s): Glissando (page 552).

Grace auto beam engraver (page 424)
   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: beam-forbid-event (page 49),
   Properties (read)
      autoBeaming (boolean)
      If set to true then beams are generated automatically.

   This engraver creates the following layout object(s): Beam (page 495).

Grace beam engraver (page 425)
   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: beam-event (page 49),
   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): Beam (page 495).

Grace_engraver (page 425)
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.

Grob_pq_engraver (page 426)
Administrative 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.).

Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

instrumentCueName (markup)
The name to print if another instrument is to be taken.
This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),
This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

Ligature_bracket_engraver (page 431)
Handle Ligature_events by engraving Ligature brackets.
Music types accepted: ligature-event (page 52),
This engraver creates the following layout object(s): LigatureBracket (page 573).

Multi_measure_rest_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 MultiMeasureRest, page 587.
Music types accepted: multi-measure-articulation-event (page 53),
multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)

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

  internalBarNumber (integer)
  Contains the current bar number. This property is used for internal time-
  keeping, among others by the Accidental_ engraver.

  measureStartNow (boolean)
  True at the beginning of a measure.

  restNumberThreshold (number)
  If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587),
MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and
MultiMeasureRestText (page 592).

New_fingering_ engraver (page 437)
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): Fingering (page 545), Script
(page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_ engraver (page 438)
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): VoiceFollower (page 670).

Note_heads_ engraver (page 438)
Generate note heads.
Music types accepted: note-event (page 54),
Properties (read)

  middleCPosition (number)
  The place of the middle C, measured in half staff-spaces. Usually deter-
mined by looking at middleCClefPosition and middleCOffset.
staffLineLayoutFunction (procedure)
Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): NoteHead (page 596).

Note_spacing_engraver (page 439)
Generate NoteSpacing, an object linking horizontal lines for use in spacing.
This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.
Music types accepted: note-event (page 54), and part-combine-event (page 54),
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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)
  countPercentRepeats (boolean)
  If set, produce counters for percent repeats.

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

  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).
Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
Print the bracketed note head after a note head with trill.
This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).

Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).
Slur_engraver (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Text_engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).

Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and
TieColumn (page 655).

Trill_spanner_engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Chapter 2: Translation

Properties (read)

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

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): TrillSpanner (page 661).

Tuplet_engraver (page 455)
   Catch tuplet events and generate appropriate bracket.
   Music types accepted: tuplet-span-event (page 58),
   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): TupletBracket (page 662), and TupletNumber (page 664).

2.1.17 InternalGregorianStaff

An internal Staff type with settings shared by multiple ancient notation schemes.

   This context creates the following layout object(s): Accidental (page 475),
   AccidentalCautionary (page 476), AccidentalPlacement (page 477),
   AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491),
   BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492),
   BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), Divisio (page 528), DotColumn (page 531), FingeringColumn (page 547),
   InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565),
   LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599),
   PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614),
   ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624),
   StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632),
   StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642),
   TimeSignature (page 655), UnaCordaPedal (page 665), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

   This context sets the following properties:
   • Set context property autoAccidentals to:
     
   • Set context property autoCautionaries to '()'.
   • Set context property caesuraTypeTransform to caesura-to-bar-line-or-divisio.
   • Set context property caesuraType to:
     
   • Set context property createSpacing to #t.
   • Set context property doubleRepeatBarType to "||".
• Set context property `doubleRepeatSegnoBarType` to "S-||".
• Set context property `endRepeatBarType` to "||".
• Set context property `endRepeatSegnoBarType` to "S-||".
• Set context property `extraNatural` to #f.
• Set context property `fineBarType` to "||".
• Set context property `fineSegnoBarType` to "S-||".
• Set context property `fineStartRepeatSegnoBarType` to "S-||".
• Set context property `forbidBreakBetweenBarLines` to #f.
• Set context property `ignoreFiguredBassRest` to #f.
• Set context property `instrumentName` to '()'.
• Set context property `localAlterations` to '()'.
• Set context property `measureBarType` to '()'.
• Set context property `ottavationMarkups` to:
  
  '((-4 . "29")
  (3 . "22")
  (2 . "15")
  (1 . "8")
  (-1 . "8")
  (-2 . "15")
  (-3 . "22")
  (-4 . "29"))

• Set context property `printKeyCancellation` to #f.
• Set context property `sectionBarType` to "||".
• Set context property `segnoBarType` to "S-||".
• Set context property `shortInstrumentName` to '()'.
• Set context property `startRepeatBarType` to "||".
• Set context property `startRepeatSegnoBarType` to "S-||".
• Set context property `underlyingRepeatBarType` to "||".
• Set grob property `extra-spacing-height` in `BreathingSign` (page 502), to item::extra-spacing-height-including-staff.
• Set grob property `extra-spacing-width` in `BreathingSign` (page 502), to :
  '
  '(-1.0 . 0.0)

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

Context `InternalGregorianStaff` can contain `CueVoice` (page 97), and `NullVoice` (page 228).

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

`Accidental_engraver` (page 400)

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.
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 bar number. 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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), and AccidentalSuggestion (page 478).

Alteration_glyph_engraver (page 401)

Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.

Properties (read)

alterationGlyphs (list)

A list mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group_engraver (page 403)

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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)

Create bar lines for various commands, including \\bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

caesuraType (list)

An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...))
specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:\ldots:’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:\ldots:’. 

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:\ldots:’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:\ldots:’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘:\ldots:’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘:\ldots:’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘:\ldots:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ‘(command args\ldots)’, but a command with no arguments
may be abbreviated to a symbol; e.g., '(()(start-repeat)) may be given as '(start-repeat).

end-repeat
   End a repeated section.

start-repeat
   Start a repeated section.

volta text
   If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
   Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
   Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
   A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
   Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
   Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
   The current bar line type, or ‘() if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

currentBarLine (graphical (layout) object)
   Set to the BarLine that Bar_engraver has created in the current timestep.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Clef_engraver (page 413)
   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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.
forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513),
CueClef (page 521), and CueEndClef (page 524).

Divisio_engraver (page 417)
Create divisiones: chant notation for points of breathing or caesura.
Music types accepted: caesura-event (page 50), fine-event (page 51),
section-event (page 55), volta-repeat-end-event (page 59), and
volta-repeat-start-event (page 59),

Properties (read)
caesuraType (list)
An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)  
(underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at
\caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-
bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of
the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional
entry (articulations . symbol-list) identifying the articulations at-
tached to the caesura in the music. If the transform function returns this
second argument unmodified, it is as if no transform function were set;
the function is free to return a different value. The transform function
can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the en-
graver has observed. bar-line indicates that the engraver has observed
a BarLine at the current moment.

This engraver creates the following layout object(s): Divisio (page 528).

Dot_column_engraver (page 418)
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): DotColumn (page 531).

Figured_bass_engraver (page 421)
Make figured bass numbers.
Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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): BassFigure (page 491), BassFigureAlignment (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.

This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that will fix the collisions.

This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.

Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

busyGros (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)

busyGros (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.).
Instrument_name_engraver (page 427)
 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 items (clef, key signature, etc.).

  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): InstrumentName (page 559).

Key_engraver (page 429)
 Engrave a key signature.
 Music types accepted: key-change-event (page 52),
 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.

  forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly re-
    quested by the user.

  forceBreak (boolean)
    Set to #t when an event forcing a line break was heard.

  keyAlterationOrder (list)
    A list of pairs that defines in what order alterations should be printed.
    The format of an entry is (step . alter), where step is a number from
    0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
    rationals for alterations in between, e.g., 1/2 for sharp.

  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): KeyCancellation (page 563), and KeySignature (page 565).

Ledger_line_engraver (page 431)
    Create the spanner to draw ledger lines, and notices objects that need ledger lines.
    This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Merge_mmrest_numbers_engraver (page 435)
    Engraver to merge multi-measure rest numbers in multiple voices.
    This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Non_musical_script_column_engraver (page 437)
    Find potentially colliding non-musical scripts and put them into a ScriptColumn object; that will fix the collisions.
    This engraver creates the following layout object(s): ScriptColumn (page 614).

Ottava_spanner_engraver (page 439)
    Create a text spanner when the ottavation property changes.
    Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48), Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
Handle collisions of rests.
Properties (read)

busyGrots (list)
A queue of \texttt{(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): \texttt{RestCollision} (page 612).

\texttt{Script\_row\_engraver} (page 445)
\begin{itemize}
\item Determine order in horizontal side position elements.
\end{itemize}
This engraver creates the following layout object(s): \texttt{ScriptRow} (page 614).

\texttt{Separating\_line\_group\_engraver} (page 445)
\begin{itemize}
\item Generate objects for computing spacing parameters.
\end{itemize}

Properties (read)

\begin{itemize}
\item \texttt{createSpacing} (boolean)
Create \texttt{StaffSpacing} objects? Should be set for staves.
\end{itemize}

Properties (write)

\begin{itemize}
\item \texttt{hasStaffSpacing} (boolean)
True if \texttt{currentCommandColumn} contains items that will affect spacing.
\end{itemize}
This engraver creates the following layout object(s): \texttt{StaffSpacing} (page 632).

\texttt{Skip\_typesetting\_engraver} (page 446)
\begin{itemize}
\item Create a \texttt{StaffEllipsis} when \texttt{skipTypesetting} is used.
\end{itemize}

Properties (read)

\begin{itemize}
\item \texttt{skipTypesetting} (boolean)
If true, no typesetting is done, speeding up the interpretation phase.
Useful for debugging large scores.
\end{itemize}
This engraver creates the following layout object(s): \texttt{StaffEllipsis} (page 628).

\texttt{Staff\_collecting\_engraver} (page 448)
\begin{itemize}
\item Maintain the \texttt{stavesFound} variable.
\end{itemize}

Properties (read)

\begin{itemize}
\item \texttt{stavesFound} (list of grobs)
A list of all staff-symbols found.
\end{itemize}

Properties (write)

\begin{itemize}
\item \texttt{stavesFound} (list of grobs)
A list of all staff-symbols found.
\end{itemize}

\texttt{Staff\_highlight\_engraver} (page 449)
\begin{itemize}
\item Highlights music passages.
\end{itemize}
Music types accepted: \texttt{staff-highlight-event} (page 56),

Properties (read)

\begin{itemize}
\item \texttt{currentCommandColumn} (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature,
etc.).
\end{itemize}
This engraver creates the following layout object(s): \texttt{StaffHighlight} (page 631).
Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),
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, '\(4 \cdot 4\)
is a 4/4 time signature.

This engraver creates the following layout object(s): TimeSignature (page 655).

2.1.18 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 (page 287).

This context creates the following layout object(s): Accidental (page 475),
AccidentalCautionary (page 476), AccidentalPlacement (page 477),
AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492),
BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), BreathingSign (page 502), CaesuraScript (page 504), Clef (page 510),
ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565), LedgerLineSpanner (page 570), NoteCollision (page 595), OttawaBracket (page 599), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628),
StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632),
SustainPedal (page 641), SustainPedallineSpanner (page 642), UnaCordaPedal (page 665),
UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set context property autoAccidentals to:
  \'(Staff #<procedure 5575a3860880 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
    #<procedure neo-modern-accidental-rule (context pitch barnum)>)
- Set context property autoCautionaries to '\(\).
- Set context property caesuraType to:
  '\(\text{'(bar-line . ",")}\)
- Set context property clefGlyph to "clefs.kievan.do".
- Set context property clefPosition to 0.
- Set context property clefTransposition to 0.
• Set context property `createSpacing` to `#t`.
• Set context property `doubleRepeatBarType` to "k".
• Set context property `endRepeatBarType` to "k".
• Set context property `extraNatural` to `#f`.
• Set context property `fineBarType` to "k".
• Set context property `forbidBreakBetweenBarLines` to `#f`.
• Set context property `ignoreFiguredBassRest` to `#f`.
• Set context property `instrumentName` to `'()``.
• Set context property `localAlterations` to `'()``.
• Set context property `measureBarType` to `'()``.
• Set context property `middleCClefPosition` to 0.
• Set context property `middleCPosition` to 0.
• Set context property `ottavationMarkups` to:
  
  '(((4 . "29")
      (3 . "22")
      (2 . "15")
      (1 . "8")
      (-1 . "8")
      (-2 . "15")
      (-3 . "22")
      (-4 . "29"))
• Set context property `printKeyCancellation` to `#f`.
• Set context property `sectionBarType` to "k".
• Set context property `shortInstrumentName` to `'()``.
• Set context property `startRepeatBarType` to "k".
• Set context property `underlyingRepeatBarType` to "k".
• Set grob property `thick-thickness` in `BarLine` (page 486), to 3.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type `KievanVoice` (page 189).

Context `KievanStaff` can contain `CueVoice` (page 97), `KievanVoice` (page 189), and `NullVoice` (page 228).

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

`Accidental_engraver` (page 400)
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.

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 bar number. This property is used for internal time-keeping, 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): Accidental (page 475),
AccidentalCautionary (page 476), AccidentalPlacement (page 477), and
AccidentalSuggestion (page 478).

**Alteration_glyph_engraver (page 401)**

Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.

Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

**Axis_group_engraver (page 403)**

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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

**Bar_engraver (page 404)**

Create bar lines for various commands, including \\bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50),
coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51),
section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

caesuraType (list)
An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-
bare line has lower priority than a measure bar line.
caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:...’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.|:’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.|’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.|’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.|:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ’(command args...)’, but a command with no arguments may be abbreviated to a symbol; e.g., ’((start-repeat))’ may be given as ’(start-repeat).

end-repeat
End a repeated section.
start-repeat
  Start a repeated section.

volta text
  If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
  Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
  Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
  A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
  Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
  Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
  Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
  The current bar line type, or ‘()’ if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

  currentBarLine (graphical (layout) object)
    Set to the BarLine that Bar_engraver has created in the current timestep.

  forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
  Notate a short break in sound that does not shorten the previous note.

Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.

If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),
Properties (read)

**breathMarkDefinitions (list)**
The description of breath marks. This is used by the `Breathing_sign_engraver`. See `scm/breath.scm` for more information.

**caesuraType (list)**
An alist

```lisp
((bar-line . bar-type)
(breath . breath-type)
(scripts . script-type...)
(underlying-bar-line . bar-type))
```

specifying which breath mark, bar line, and scripts to create at a caesura. All entries are optional.

*bar-line* has higher priority than a measure bar line and *underlying-bar-line* has lower priority than a measure bar line.

**caesuraTypeTransform (procedure)**
An engraver callback taking three arguments and returning an alist of the same kind as *caesuraType*.

The first argument is the context.

The second argument is the value of *caesuraType* with an additional entry `articulations . symbol-list` identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. *bar-line* indicates that the engraver has observed a BarLine at the current moment.

**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): `BreathingSign` (page 502), and `CaesuraScript` (page 504).

**Clef_engraver (page 413)**
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.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly re-
   quested by the user.

forceBreak (boolean)
   Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and
ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly re-
   quested by the user.

forceBreak (boolean)
   Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513),
CueClef (page 521), and CueEndClef (page 524).
Chapter 2: Translation

Dot_column_engraver (page 418)
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): DotColumn (page 531).

Figured_bass_engraver (page 421)
Make figured bass numbers.
Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.
This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that will fix the collisions.
This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)
fontSize (number)
The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
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)

busyGros (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
eads, spanners, etc.).

Instrument_name_ENGRAVER (page 427)
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 items (clef, key signature,
etc.).

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): InstrumentName (page 559).

Key_ENGRAVER (page 429)
Engrave a key signature.

Music types accepted: key-change-event (page 52),

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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

keyAlterationOrder (list)
A list of pairs that defines in what order alterations should be printed.
The format of an entry is (step . alter), where step is a number from
0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
rationals for alterations in between, e.g., 1/2 for sharp.
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): KeyCancellation (page 563), and KeySignature (page 565).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines. This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices. This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn object, that will fix the collisions. This engraver creates the following layout object(s): ScriptColumn (page 614).

Ottava_spanner_engraver (page 439)
Create a text spanner when the ottavation property changes.

Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),
Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.
Properties (read)
currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature,
etc.).

This engraver creates the following layout object(s): SostenutoPedalLineSpanner
(page 624), SustainPedalLineSpanner (page 642), and
UnaCordaPedalLineSpanner (page 666).
Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and
una-corda-event (page 58),
Properties (read)
currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature,
etc.).
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): PianoPedalBracket
(page 606), SostenutoPedal (page 623), SustainPedal (page 641), and
UnaCordaPedal (page 665).
Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
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): RestCollision (page 612).

Script_row_engraver (page 445)
Determine order in horizontal side position elements.
This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
Generate objects for computing spacing parameters.
Properties (read)
  createSpacing (boolean)
    Create StaffSpacing objects? Should be set for staves.
Properties (write)
  hasStaffSpacing (boolean)
    True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
Create a StaffEllipsis when skipTypesetting is used.
Properties (read)
  skipTypesetting (boolean)
    If true, no typesetting is done, speeding up the interpretation phase.
    Useful for debugging large scores.

This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature,
    etc.).
Chapter 2: Translation

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

2.1.19 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 (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Fingering (page 545), Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), KievanLigature (page 568), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), Stem (page 634), StemStub (page 636), StemTremolo (page 637), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), and VoiceFollower (page 670).

This context sets the following properties:

- Set context property alterationGlyphs to:
  '\((-1/2 . "accidentals.kievanM1")
   (1/2 . "accidentals.kievan1")
  )'
- Set context property autoBeaming to #f.
- Set grob property duration-log in NoteHead (page 596), to note-head::calc-kievan-duration-log.
- Set grob property length in Stem (page 634), to 0.0.
- Set grob property positions in Beam (page 495), to beam::get-kievan-positions.
- Set grob property quantized-positions in Beam (page 495), to beam::get-kievan-quantized-positions.
- Set grob property stencil in Flag (page 547), to #f.
- Set grob property stencil in Slur (page 621), to #f.
- Set grob property style in Dots (page 531), to 'kievan.
- Set grob property style in NoteHead (page 596), to 'kievan.
- Set grob property style in Rest (page 611), to 'mensural.
- Set grob property X-offset in Stem (page 634), to stem::kievan-offset-callback.
Chapter 2: Translation

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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.
Music types accepted: arpeggio-event (page 49),
This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties
stemLeftBeamCount and stemRightBeamCount.
Music types accepted: beam-forbid-event (page 49),
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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags
instead of beams.
Music types accepted: beam-event (page 49),
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.
Chapter 2: Translation

This engraver creates the following layout object(s): Beam (page 495).

Bend_ engraver (page 409)
Create fall spanners.
Music types accepted: bend-after-event (page 49),
Properties (read)

- currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.

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

- 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): BendAfter (page 497).

Breathing_sign_ engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)

- breathMarkType (symbol)
  The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_ engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_ engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514),
and ClusterSpannerBeacon (page 515).

Dots_ engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_ engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly requested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner (page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56),

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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),

This engraver creates the following layout object(s): FingerGlideSpanner (page 543).
Fingering_ engraver (page 422)
  Create fingering scripts.
  Music types accepted: fingering-event (page 51),
  This engraver creates the following layout object(s): Fingering (page 545).

Font_size_ engraver (page 422)
  Put fontSize into font-size grob property.
  Properties (read)
    fontSize (number)
      The relative size of all grobs in a context.

Forbid_line_break_ engraver (page 423)
  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, except if explicitly re-
      quested by the user.

Glissando_ engraver (page 424)
  Engrave glissandi.
  Music types accepted: glissando-event (page 51),
  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 nu-
      mber of note-heads in the two note columns between which the glissandi
      occur.
  This engraver creates the following layout object(s): Glissando (page 552).

Grace_auto_beam_ engraver (page 424)
  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: beam-forbid-event (page 49),
  Properties (read)
    autoBeaming (boolean)
      If set to true then beams are generated automatically.
  This engraver creates the following layout object(s): Beam (page 495).

Grace_beam_ engraver (page 425)
  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: beam-event (page 49),
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): Beam (page 495).

Grace_engraver (page 425)
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.

Grob_pq_engraver (page 426)
Administrates 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.).

Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.

This engraver is deprecated.

Properties (read)

instrumentCueName (markup)
The name to print if another instrument is to be taken.

This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Kievan_ligature_engraver (page 430)
Handle Kievan_ligature_events by glueing Kievan heads together.

Music types accepted: ligature-event (page 52),

This engraver creates the following layout object(s): KievanLigature (page 568).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),

This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

**Multi_measure_rest_engraver** (page 436)

Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.

Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53).

Properties (read)

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

- **internalBarNumber** (integer)
  Contains the current bar number. This property is used for internal time-keeping, among others by the Accidental_engraver.

- **measureStartNow** (boolean)
  True at the beginning of a measure.

- **restNumberThreshold** (number)
  If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and MultiMeasureRestText (page 592).

**New_fingering_engraver** (page 437)

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): Fingering (page 545), Script (page 613), StringNumber (page 638), and StrokeFinger (page 640).

**Note_head_line_engraver** (page 438)

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): VoiceFollower (page 670).
Note_heads_engraver (page 438)
Generate note heads.
Music types accepted: note-event (page 54),
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): NoteHead (page 596).

Note_spacing_engraver (page 439)
Generate NoteSpacing, an object linking horizontal lines for use in spacing.
This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.
Music types accepted: note-event (page 54), and part-combine-event (page 54),
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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)

countPercentRepeats (boolean)
If set, produce counters for percent repeats.

currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).
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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
    Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
    Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),

This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
    Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
    Create repeat ties.

This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
    Engrave rests.

Music types accepted: rest-event (page 55),

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): Rest (page 611).

Rhythmic_column_engraver (page 445)
    Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
    Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.

This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
    Handle note scripted articulations.

Music types accepted: articulation-event (page 49),

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): Script (page 613).

**Slash_repeat_engraver** (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

**Slur_engraver** (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

**Spanner_break_forbid_engraver** (page 448)
Forbid breaks in certain spanners.

**Stem_engraver** (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)
   - currentBarLine (graphical (layout) object)
     Set to the BarLine that Bar_engraver has created in the current timestep.
   - 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.
This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

**Text_engraver** (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

**Text_spanner_engraver** (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
Properties (read)
   - currentMusicalColumn (graphical (layout) object)
     Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).
Chapter 2: Translation

This engraver creates the following layout object(s): TextSpanner (page 652).

**Tie_engraver** (page 452)
Generate ties between note heads of equal pitch.

**Music types accepted:** tie-event (page 58),

**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): Tie (page 653), and
TieColumn (page 655).

**Trill_spanner_engraver** (page 455)
Create trill spanners.

**Music types accepted:** trill-span-event (page 58),

**Properties (read)**

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

- **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): TrillSpanner (page 661).

**Tuplet_engraver** (page 455)
Catch tuplet events and generate appropriate bracket.

**Music types accepted:** tuplet-span-event (page 58),

**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 signa-
  tures, etc.) before the note.

This engraver creates the following layout object(s): TupletBracket (page 662), and
TupletNumber (page 664).

### 2.1.20 Lyrics

Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

This context creates the following layout object(s): InstrumentName (page 559),
LyricExtender (page 574), LyricHyphen (page 575), LyricSpace (page 578), LyricText
This context sets the following properties:

- Set context property `instrumentName` to `()`. 
- Set context property `lyricRepeatCountFormatter` to `#<procedure 5575a3702120 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:210:4 (context repeat-count)>`. 
- Set context property `searchForVoice` to `#f`. 
- Set context property `shortInstrumentName` to `()`. 
- Set grob property `bar-extent` in `BarLine` (page 486), to: `'(-0.05 . 0.05)` 
- Set grob property `font-size` in `InstrumentName` (page 559), to 1.0. 
- Set grob property `nonstaff-nonstaff-spacing` in `VerticalAxisGroup` (page 668), to: `'((basic-distance . 0) (minimum-distance . 2.8) (padding . 0.2) (stretchability . 0))` 
- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 668), to: `'((basic-distance . 5.5) (padding . 0.5) (stretchability . 1))` 
- Set grob property `nonstaff-unrelatedstaff-spacing.padding` in `VerticalAxisGroup` (page 668), to 1.5. 
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 668), to `#t`. 
- Set grob property `remove-first` in `VerticalAxisGroup` (page 668), to `#t`. 
- Set grob property `self-alignment-Y` in `InstrumentName` (page 559), to `#f`. 
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 668), 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):

```
Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).
- `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): \texttt{VerticalAxisGroup} (page 668).

\textbf{Extender\_engraver} (page 421)

Create lyric extenders.

Music types accepted: \texttt{compleatize-extender-event} (page 50), and \texttt{extender-event} (page 51),

Properties (read)

\begin{itemize}
  \item \texttt{extendersOverRests} (boolean)
    \begin{itemize}
      \item Whether to continue extenders as they cross a rest.
    \end{itemize}
\end{itemize}

This engraver creates the following layout object(s): \texttt{LyricExtender} (page 574).

\textbf{Font\_size\_engraver} (page 422)

Put \texttt{fontSize} into \texttt{font-size} grob property.

Properties (read)

\begin{itemize}
  \item \texttt{fontSize} (number)
    \begin{itemize}
      \item The relative size of all grobs in a context.
    \end{itemize}
\end{itemize}

\textbf{Hyphen\_engraver} (page 427)

Create lyric hyphens, vowel transitions and distance constraints between words.

Music types accepted: \texttt{hyphen-event} (page 52), and \texttt{vowel-transition-event} (page 59),

This engraver creates the following layout object(s): \texttt{LyricHyphen} (page 575), \texttt{LyricSpace} (page 578), and \texttt{VowelTransition} (page 673).

\textbf{Instrument\_name\_engraver} (page 427)

Create a system start text for instrument or vocal names.

Properties (read)

\begin{itemize}
  \item \texttt{currentCommandColumn} (graphical (layout) object)
    \begin{itemize}
      \item Grob that is X-parent to all current breakable items (clef, key signature, etc.).
    \end{itemize}
  \item \texttt{instrumentName} (markup)
    \begin{itemize}
      \item The name to print left of a staff. The \texttt{instrumentName} property labels the staff in the first system, and the \texttt{shortInstrumentName} property labels following lines.
    \end{itemize}
  \item \texttt{shortInstrumentName} (markup)
    \begin{itemize}
      \item See \texttt{instrumentName}.
    \end{itemize}
  \item \texttt{shortVocalName} (markup)
    \begin{itemize}
      \item Name of a vocal line, short version.
    \end{itemize}
  \item \texttt{vocalName} (markup)
    \begin{itemize}
      \item Name of a vocal line.
    \end{itemize}
\end{itemize}

This engraver creates the following layout object(s): \texttt{InstrumentName} (page 559).

\textbf{Lyric\_engraver} (page 431)

Engrave text for lyrics.

Music types accepted: \texttt{lyric-event} (page 52),

Properties (read)

\begin{itemize}
  \item \texttt{ignoreMelismata} (boolean)
    \begin{itemize}
      \item Ignore melismata for this Section "Lyrics" in \textit{Internals Reference} line.
    \end{itemize}
\end{itemize}
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): LyricText (page 579).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Stanza_number_engraver (page 450)
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): StanzaNumber (page 633).

2.1.21 MensuralStaff

Configure division commands such as \section to create Divisio grobs rather than BarLine grobs. This does not affect measure bar lines or the properties of the grobs themselves.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), Custos (page 527), Divisio (page 528), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565), LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642), TimeSignature (page 655), UnaCordaPedal (page 665), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set context property alterationGlyphs to:
  
  '(((-1/2 . "accidentals.mensuralM1")
  (0 . "accidentals.vaticana0")
  (1/2 . "accidentals.mensural1"))

- Set context property autoAccidentals to:
  
  '((Staff #<procedure 5575a36e3080 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
  • Set context property autoCautionaries to '()
  • Set context property caesuraTypeTransform to caesura-to-bar-line-or-divisio.
  • Set context property caesuraTypeTransform to caesura-to-divisio.
  • Set context property caesuraType to:
    
    '((breath . varcomma))
• Set context property clefGlyph to "clefs.mensural.g".
• Set context property clefPosition to -2.
• Set context property clefTransposition to 0.
• Set context property createSpacing to #t.
• Set context property doubleRepeatBarType to "||".
• Set context property doubleRepeatBarType to '('.
• Set context property doubleRepeatSegnoBarType to "S-||".
• Set context property doubleRepeatSegnoBarType to "S-||".
• Set context property endRepeatBarType to "||".
• Set context property endRepeatBarType to '().
• Set context property endRepeatSegnoBarType to "S-||".
• Set context property endRepeatSegnoBarType to "S-||".
• Set context property extraNatural to #f.
• Set context property fineBarType to "".
• Set context property fineBarType to "||".
• Set context property fineSegnoBarType to "S-||".
• Set context property fineSegnoBarType to "S-||".
• Set context property fineStartRepeatSegnoBarType to "S-||".
• Set context property fineStartRepeatSegnoBarType to "S-||".
• Set context property forbidBreakBetweenBarLines to #f.
• Set context property ignoreFiguredBassRest to #f.
• Set context property instrumentName to '().
• Set context property localAlterations to '().
• Set context property measureBarType to '().
• Set context property middleCClefPosition to -6.
• Set context property middleCPosition to -6.
• Set context property ottavationMarkups to:
  '((4 . "29")
   (3 . "22")
   (2 . "15")
   (1 . "8")
   (-1 . "8")
   (-2 . "15")
   (-3 . "22")
   (-4 . "29")
   )
• Set context property printKeyCancellation to #f.
• Set context property sectionBarType to "".
• Set context property sectionBarType to "||".
• Set context property segnoBarType to "S-||".
• Set context property segnoBarType to "S-||".
• Set context property shortInstrumentName to '().
• Set context property startRepeatBarType to "||".
• Set context property startRepeatBarType to '().
• Set context property startRepeatSegnoBarType to "S-||".
• Set context property `startRepeatSegnoBarType` to "S-||".
• Set context property `underlyingRepeatBarType` to "".
• Set context property `underlyingRepeatBarType` to "||".
• Set grob property `extra-spacing-height` in `BreathingSign` (page 502), to item::extra-spacing-height-including-staff.
• Set grob property `extra-spacing-width` in `BreathingSign` (page 502), to: 
  '(-1.0 . 0.0)
• Set grob property `font-size` in `BreathingSign` (page 502), to -2.
• Set grob property `font-size` in `Divisio` (page 528), to -2.
• Set grob property `hair-thickness` in `BarLine` (page 486), to 0.6.
• Set grob property `neutral-direction` in `Custos` (page 527), to -1.
• Set grob property `neutral-position` in `Custos` (page 527), to 3.
• Set grob property `style` in `Custos` (page 527), to 'mensural.'
• Set grob property `style` in `TimeSignature` (page 655), to 'mensural.'
• Set grob property `thick-thickness` in `BarLine` (page 486), to 1.8.
• Set grob property `thick-thickness` in `BreathingSign` (page 502), to 1.
• Set grob property `thick-thickness` in `Divisio` (page 528), to 1.
• Set grob property `thick-thickness` in `StaffSymbol` (page 632), to 0.6.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type `MensuralVoice` (page 216).

Context `MensuralStaff` can contain `CueVoice` (page 97), `MensuralVoice` (page 216), and `NullVoice` (page 228).

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

*Accidental_engraver* (page 400)

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidentals 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.

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 bar number. This property is used for internal time-keeping, 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): `Accidental` (page 475), `AccidentalCautionary` (page 476), `AccidentalPlacement` (page 477), and `AccidentalSuggestion` (page 478).

### Alteration_glyph_engraver (page 401)

Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context's alterationGlyphs property, when defined.

Properties (read)

- **alterationGlyphs** (list)
  - Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.
Axis_group engraver (page 403)
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 items (clef, key signature, etc.).

  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): VerticalAxisGroup (page 668).

Bar engraver (page 404)
Create bar lines for various commands, including \bar.
If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.
Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59),
Properties (read)

  caesuraType (list)
    An alist
      ((bar-line . bar-type)
       (breath . breath-type)
       (scripts . script-type...) )
      (underlying-bar-line . bar-type))
    specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.
    bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.
doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:...:’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.S.|.’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.S’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.|.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.|.S:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ‘(command args...)’, but a command with no arguments may be abbreviated to a symbol; e.g., ‘((start-repeat))’ may be given as ‘(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.
startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.|.’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a
\repeat volta. The default is ‘S.|.’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line
would normally appear, for example at the end of a system broken in
mid measure where the next system begins with a segno. Where there is
also a repeat bar line, the repeat bar line takes precedence and this value
is appended to it as an annotation. The default is ‘||’.

whichBar (string)
The current bar line type, or ‘() if there is no bar line. Setting this ex-
plicitly in user code is deprecated. Use \bar or related commands to set
it.

Properties (write)
currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current
timestep.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Clef_engraver (page 413)
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.
forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.
forceBreak (boolean)
Set to #t when an event forcing a line break was heard.
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): Clef (page 510), and ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.

  forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly requested by the user.

  forceBreak (boolean)
    Set to #t when an event forcing a line break was heard.

  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): ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

Custos_engraver (page 417)
Engrave custodes.
Properties (read)

  forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly requested by the user.

  forceBreak (boolean)
    Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): Custos (page 527).
**Divisio_engraver** (page 417)
Create divisiones: chant notation for points of breathing or caesura.

Music types accepted: caesura-event (page 50), fine-event (page 51), section-event (page 55), volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59),

Properties (read)

```lisp
caesuraType (list)
   An alist
   ((bar-line . bar-type)
    (breath . breath-type)
    (scripts . script-type...)
    (underlying-bar-line . bar-type))
```

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

- bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

```lisp
caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
   The first argument is the context.
   The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
   The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.
```

This engraver creates the following layout object(s): Divisio (page 528).

**Dot_column_engraver** (page 418)
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): DotColumn (page 531).

**Figured_bass_engraver** (page 421)
Make figured bass numbers.

Music types accepted: bass-figure-event (page 49), and rest-event (page 55),

Properties (read)

```lisp
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): BassFigure (page 491), BassFigureAlignment (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
   Position figured bass alignments over notes.
   This engraver creates the following layout object(s):
   BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
   Find potentially colliding scripts and put them into a FingeringColumn object; that will fix the collisions.
   This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
   Put fontSize into font-size grob property.
   Properties (read)
      fontSize (number)
      The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
   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.).

Instrument_name_engraver (page 427)
   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 items (clef, key signature, etc.).

      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): InstrumentName (page 559).

Key_engraver (page 429)
Engrave a key signature.

Music types accepted: key-change-event (page 52),

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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

keyAlterationOrder (list)
A list of pairs that defines in what order alterations should be printed.
The format of an entry is (step . alter), where step is a number from
0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
rationals for alterations in between, e.g., 1/2 for sharp.

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): KeyCancellation (page 563), and KeySignature (page 565).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines.
This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Ottava_spanner_engraver (page 439)
Create a text spanner when the ottavation property changes.
Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.

Properties (read)

currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).
This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

**Piano_pedal_engraver (page 442)**
Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58).

Properties (read)

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

- `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): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

**Pure_from_neighbor_engraver (page 443)**
Coordinates items that get their pure heights from their neighbors.

**Rest_collision_engraver (page 444)**
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): RestCollision (page 612).

**Script_row_engraver (page 445)**
Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 614).

**Separating_line_group_engraver (page 445)**
Generate objects for computing spacing parameters.
Properties (read)
  createSpacing (boolean)
      Create StaffSpacing objects? Should be set for staves.

Properties (write)
  hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
Create a StaffEllipsis when skipTypesetting is used.

Properties (read)
  skipTypesetting (boolean)
      If true, no typesetting is done, speeding up the interpretation phase.
      Useful for debugging large scores.

This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),

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): TimeSignature (page 655).

2.1.22 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 (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam
   (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner
   (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots
   (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533),
   DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539),
   DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Fingering (page 545),
   Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560),
   LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), MensuralLigature
   (page 585), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589),
   MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn
   (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602),
   PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609),
   RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613),
   ScriptColumn (page 614), Stem (page 634), StemStub (page 636), StemTremolo (page 637),
   StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner
   (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657),
   TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses
   (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664),
   and VoiceFollower (page 670).

This context sets the following properties:
   • Set context property autoBeaming to #f.
   • Set grob property style in Flag (page 547), to 'mensural.
   • Set grob property style in NoteHead (page 596), to 'mensural.
   • Set grob property style in Rest (page 611), to 'mensural.

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):

Arpeggio_engraver (page 402)
   Generate an Arpeggio symbol.

   Music types accepted: arpeggio-event (page 49),

   This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
   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 2.2.141 [Stem_engraver], page 450, properties
   stemLeftBeamCount and stemRightBeamCount.

   Music types accepted: beam-forbid-event (page 49),
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

beatStructure (list)
List of baseMoments that are combined to make beats.

subdivideBeams (boolean)
If set, multiple beams will be subdivided at baseMoment positions by

This engraver creates the following layout object(s): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags

Music types accepted: beam-event (page 49),

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

This engraver creates the following layout object(s): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.

Music types accepted: bend-after-event (page 49),

Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

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

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): \texttt{BendAfter} (page 497).

\texttt{Breathing\_sign\_engraver} (page 410)
Notate breath marks.
Music types accepted: \texttt{breathing-event} (page 50),
Properties (read)
\begin{itemize}
\item \texttt{breathMarkType} (symbol)
The type of \texttt{BreathingSign} to create at \texttt{\breathe}.
\end{itemize}
This engraver creates the following layout object(s): \texttt{BreathingSign} (page 502).

\texttt{Chord\_tremolo\_engraver} (page 412)
Generate beams for tremolo repeats.
Music types accepted: \texttt{tremolo-span-event} (page 58),
This engraver creates the following layout object(s): \texttt{Beam} (page 495).

\texttt{Cluster\_spanner\_engraver} (page 413)
Engrave a cluster using \texttt{Spanner} notation.
Music types accepted: \texttt{cluster-note-event} (page 50),
This engraver creates the following layout object(s): \texttt{ClusterSpanner} (page 514),
and \texttt{ClusterSpannerBeacon} (page 515).

\texttt{Dots\_engraver} (page 418)
Create Section 3.1.43 \texttt{[Dots]}, page 531, objects for Section 3.2.118 \texttt{[rhythmic-head-interface]}, page 734.
This engraver creates the following layout object(s): \texttt{Dots} (page 531).

\texttt{Double\_percent\_repeat\_engraver} (page 418)
Make double measure repeats.
Music types accepted: \texttt{double-percent-event} (page 50),
Properties (read)
\begin{itemize}
\item \texttt{countPercentRepeats} (boolean)
If set, produce counters for percent repeats.
\item \texttt{measureLength} (moment)
Length of one measure in the current time signature.
\item \texttt{repeatCountVisibility} (procedure)
A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when \texttt{countPercentRepeats} is set.
\end{itemize}
Properties (write)
\begin{itemize}
\item \texttt{forbidBreak} (boolean)
If set to \texttt{#t}, prevent a line break at this point, except if explicitly requested by the user.
\end{itemize}
This engraver creates the following layout object(s): \texttt{DoublePercentRepeat} (page 532), and \texttt{DoublePercentRepeatCounter} (page 533).

\texttt{Dynamic\_align\_engraver} (page 419)
Align hairpins and dynamic texts on a horizontal line.
Properties (read)
\begin{itemize}
\item \texttt{currentMusicalColumn} (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).
\end{itemize}
This engraver creates the following layout object(s): DynamicLineSpanner (page 538).

**Dynamic_engraver** (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56),
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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

**Finger_glide_engraver** (page 422)
Engraver to print a line between two Fingering grobs.
Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

**Fingering_engraver** (page 422)
Create fingering scripts.
Music types accepted: fingering-event (page 51),
This engraver creates the following layout object(s): Fingering (page 545).

**Font_size_engraver** (page 422)
Put fontSize into font-size grob property.
Properties (read)

- **fontSize** (number)
  The relative size of all grobs in a context.

**Forbid_line_break_engraver** (page 423)
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, except if explicitly re-
   quested by the user.

Glissando_engraver (page 424)
   Engrave glissandi.
   Music types accepted: glissando-event (page 51),
   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 nu-
      mber of note-heads in the two note columns between which the glissandi
      occur.

   This engraver creates the following layout object(s): Glissando (page 552).

Grace_auto_beam_engraver (page 424)
   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: beam-forbid-event (page 49),
   Properties (read)

   autoBeaming (boolean)
      If set to true then beams are generated automatically.

   This engraver creates the following layout object(s): Beam (page 495).

Grace_beam_engraver (page 425)
   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: beam-event (page 49),
   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): Beam (page 495).

Grace_engraver (page 425)
   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.
Grob_pq_engraver (page 426)

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.).

Instrument_switch_engraver (page 427)

Create a cue text for taking instrument.

This engraver is deprecated.

Properties (read)

instrumentCueName (markup)
The name to print if another instrument is to be taken.

This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 52),

This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

Mensural_ligature_engraver (page 435)

Handle Mensural_ligature_events by glueing special ligature heads together.

Music types accepted: ligature-event (page 52),

This engraver creates the following layout object(s): MensuralLigature (page 585).

Multi_measure_rest_engraver (page 436)

Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.

Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),

Properties (read)

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

internalBarNumber (integer)
Contains the current barmumber. This property is used for internal time-keeping, among others by the Accidental_engraver.

measureStartNow (boolean)
True at the beginning of a measure.
restNumberThreshold (number)

If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and MultiMeasureRestText (page 592).

New_fingering_engraver (page 437)

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): Fingering (page 545), Script (page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)

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): VoiceFollower (page 670).

Note_heads_engraver (page 438)

Generate note heads.

Music types accepted: note-event (page 54),

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): NoteHead (page 596).

Note_spacing_engraver (page 439)

Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),
Part_combine_engraver (page 441)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: note-event (page 54), and part-combine-event (page 54),

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?

soloIIIText (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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)

Make whole measure repeats.

Music types accepted: percent-event (page 54),

Properties (read)

countPercentRepeats (boolean)
If set, produce counters for percent repeats.

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

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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)

Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.

Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),

This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).
Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and
RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).

Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Stem_engraver (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)
  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

Text_engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).

Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and TieColumn (page 655).

Trill_spanner_engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)

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

  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): TrillSpanner (page 661).

Tuplet_engraver (page 455)
Catch tuplet events and generate appropriate bracket.
Music types accepted: tuplet-span-event (page 58),
Properties (read)
  tupleFullLength (boolean)
  If set, the tuplet is printed up to the start of the next note.
  tupleFullLengthNote (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): TupletBracket (page 662),
and TupletNumber (page 664).

2.1.23 NoteNames
A context for printing the names of notes.
This context also accepts commands for the following context(s): Staff (page 287).
This context creates the following layout object(s): NoteName (page 597), StaffSpacing (page 632), Tie (page 653), TieColumn (page 655), and VerticalAxisGroup (page 668).
This context sets the following properties:
• Set grob property nonstaff-nonstaff-spacing in VerticalAxisGroup (page 668), to :
  '(((basic-distance . 0)
    (minimum-distance . 2.8)
    (padding . 0.2)
    (stretchability . 0))
• Set grob property nonstaff-relatedstaff-spacing in VerticalAxisGroup (page 668), to :
  '(((basic-distance . 5.5)
    (padding . 0.5)
    (stretchability . 1))
• Set grob property nonstaff-unrelatedstaff-spacing.padding in VerticalAxisGroup (page 668), to 1.5.
• Set grob property staff-affinity in VerticalAxisGroup (page 668), 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):
  Alteration_glyph_engraver (page 401)
  Set the glyph-name-alist of all grobs having the accidental-switch-interface
to the value of the context’s alterationGlyphs property, when defined.
Properties (read)
  alterationGlyphs (list)
  Alist mapping alterations to accidental glyphs. Alterations are given as
  exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print
  accidentals.
  Axis_group_engraver (page 403)
  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 items (clef, key signature, etc.).

  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): VerticalAxisGroup
           (page 668).

Note_name_engraver (page 438)
         Print pitches as words.
         Music types accepted: note-event (page 54),

Properties (read)

  noteNameFunction (procedure)
          Function used to convert pitches into strings and markups.

  noteNameSeparator (string)
          String used to separate simultaneous NoteName objects.

  printAccidentalNames (boolean or symbol)
          Print accidentals in the NoteNames context.

  printNotesLanguage (string)
          Use a specific language in the NoteNames context.

  printOctaveNames (boolean or symbol)
          Print octave marks in the NoteNames context.

This engraver creates the following layout object(s): NoteName (page 597).

Separating_line_group_engraver (page 445)
         Generate objects for computing spacing parameters.

Properties (read)

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

Properties (write)

  hasStaffSpacing (boolean)
                 True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Tie_engraver (page 452)
         Generate ties between note heads of equal pitch.
         Music types accepted: tie-event (page 58),
Chapter 2: Translation

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): Tie (page 653), and
TieColumn (page 655).

2.1.24 NullVoice

For aligning lyrics without printing notes

This context also accepts commands for the following context(s): Staff (page 287), and
Voice (page 390).

This context creates the following layout object(s): Beam (page 495), NoteHead (page 596),
Slur (page 621), Tie (page 653), and TieColumn (page 655).

This context sets the following properties:

- Set context property nullAccidentals to #t.
- Set context property squashedPosition to 0.
- Set grob property no-ledgers in NoteHead (page 596), to #t.
- Set grob property stencil in Beam (page 495), to #f.
- Set grob property stencil in NoteHead (page 596), to #f.
- Set grob property stencil in Slur (page 621), to #f.
- Set grob property stencil in Tie (page 653), to #f.
- Set grob property X-extent in NoteHead (page 596), to #<procedure 5575a37c7d60 at
ice-9/eval.scm:333:13 (a)>.

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):

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags
instead of beams.

Music types accepted: beam-event (page 49),

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): Beam (page 495).

Grob_pq_engraver (page 426)
Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

busyGrobs (list)
A queue of \((end\text{-}moment . \ grob)\) 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 \((end\text{-}moment . \ grob)\) cons cells. This is for internal (C++) use only. This property contains the grobs which are still busy (e.g., note heads, spanners, etc.).

Note_heads_engraver (page 438)
Generate note heads.

Music types accepted: note-event (page 54),

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): NoteHead (page 596).

Pitch_squash_engraver (page 443)
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.

Slur_engraver (page 446)
Build slur grobs from slur events.

Music types accepted: note-event (page 54), and slur-event (page 56),

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): Slur (page 621).
Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and
TieColumn (page 655).

2.1.25 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): VerticalAxisGroup (page 668).

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit
context of type Staff (page 287).

Context OneStaff can contain ChordNames (page 95), DrumStaff (page 108),
Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133),
GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff
(page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202),
NoteNames (page 226), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff
(page 287), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff
(page 366).

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

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).
  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): VerticalAxisGroup (page 668).

### 2.1.26 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 (page 287).

This context creates the following layout object(s): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), AccidentalSuggestion (page 477), BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), BreathingSign (page 502), CaesuraScript (page 504), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), Custos (page 527), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565), LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SigmoidRepitionis (page 618), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642), TimeSignature (page 655), UnaCordaPedal (page 665), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:
- Set context property autoAccidentals to:
  '\((Staff \langle\textit{procedure 5575a38212c0 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0}\rangle\langle\textit{procedure neo-modern-accidental-rule (context pitch barnum)}\rangle)\)'
- Set context property autoCautonaries to '\(\()\).
- Set context property clefGlyph to "clefs.petrucci.g".
- Set context property clefPosition to -2.
- Set context property clefTransposition to 0.
- Set context property createSpacing to \#t.
- Set context property doubleRepeatBarType to '\(\)\).
- Set context property endRepeatBarType to '\(\)\).
- Set context property extraNatural to \#f.
- Set context property fineBarType to "|.".
- Set context property forbidBreakBetweenBarLines to \#f.
- Set context property ignoreFiguredBassRest to \#f.
- Set context property instrumentName to '\(\)\).
- Set context property localAlterations to '\(\)\).
- Set context property measureBarType to '\(\)\).
- Set context property middleCClefPosition to -6.
- Set context property middleCPosition to -6.
- Set context property ottavationMarkups to:
  '\((4 . "29")
  (3 . "22")
  (2 . "15")
  (1 . "8")\)
• Set context property printKeyCancellation to #f.
• Set context property sectionBarType to "||".
• Set context property shortInstrumentName to '()
• Set context property startRepeatBarType to "||".
• Set context property underlyingRepeatBarType to '()
• Set grob property bar-extent in BarLine (page 486), to:
  '(-2.5 . 2.5)
• Set grob property bar-extent in SignumRepetitionis (page 618), to:
  '(-2.5 . 2.5)
• Set grob property hair-thickness in BarLine (page 486), to 2.21.
• Set grob property hair-thickness in SignumRepetitionis (page 618), to 2.21.
• Set grob property kern in BarLine (page 486), to 2.9.
• Set grob property kern in SignumRepetitionis (page 618), to 2.9.
• Set grob property neutral-direction in Custos (page 527), to -1.
• Set grob property neutral-position in Custos (page 527), to 3.
• Set grob property rounded in BarLine (page 486), to #t.
• Set grob property rounded in SignumRepetitionis (page 618), to #t.
• Set grob property style in Custos (page 527), to 'mensural.
• Set grob property thick-thickness in BarLine (page 486), to 2.9.
• Set grob property thick-thickness in SignumRepetitionis (page 618), to 2.9.
• Set grob property thickness in StaffSymbol (page 632), to 1.3.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type PetrucciVoice (page 244).

Context PetrucciStaff can contain CueVoice (page 97), NullVoice (page 228), and PetrucciVoice (page 244).

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

Accidental_engraver (page 400)

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.

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 bar number. This property is used for internal time-keeping, 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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), and AccidentalSuggestion (page 478).

**Alteration_glyph_engraver (page 401)**
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.

Properties (read)

- **alterationGlyphs (list)**
  Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

**Axis_group_engraver (page 403)**
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 items (clef, key signature, etc.).

- **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): VerticalAxisGroup (page 668).

**Bar_engraver (page 404)**
Create bar lines for various commands, including \bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

- **caesuraType (list)**
  An alist
  
  ```lisp
  ((bar-line . bar-type)
   (breath . breath-type)
   (scripts . script-type...)
   (underlying-bar-line . bar-type))
  ```

  specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

  bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.
caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:::’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘::|:.’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘|.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘|.S|.’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.S’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.S|:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ‘((command args...))’, but a command with no arguments may be abbreviated to a symbol; e.g., ‘((start-repeat))’ may be given as ‘(start-repeat).

end-repeat
End a repeated section.
start-repeat
  Start a repeated section.

volta text
  If text is markup, start a volta bracket with that label; if text is #f,
  end a volta bracket.

sectionBarType (string)
  Bar line to insert at \section. Where there is also a repeat bar line, the
  repeat bar line takes precedence and this value is appended to it as an
  annotation. The default is ‘||’.

segnoBarType (string)
  Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
  A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
  Bar line to insert at the start of a \repeat volta. The default is ‘.:’.

startRepeatSegnoBarType (string)
  Bar line to insert where an in-staff segno coincides with the start of a
  \repeat volta. The default is ‘S.:’.

underlyingRepeatBarType (string)
  Bar line to insert at points of repetition or departure where no bar lin
  e would normally appear, for example at the end of a system broken in
  mid measure where the next system begins with a segno. Where there is
  also a repeat bar line, the repeat bar line takes precedence and this value
  is appended to it as an annotation. The default is ‘||’.

whichBar (string)
  The current bar line type, or ‘()’ if there is no bar line. Setting this ex-
  plicitly in user code is deprecated. Use \bar or related commands to set
  it.

Properties (write)

  currentBarLine (graphical (layout) object)
    Set to the BarLine that Bar_engraver has created in the current
    timestep.

  forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly re-
    quested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
  Notate a short break in sound that does not shorten the previous note.
  Depending on the result of passing the value of caesuraType through
  caesuraTypeTransform, this engraver may create a BreathingSign with
  CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align
  them to a BarLine.
  If this engraver observes a BarLine, it calls caesuraTypeTransform again with the
  new information, and if necessary, recreates its grobs.
  Music types accepted: caesura-event (page 50),
Properties (read)

breathMarkDefinitions (list)

The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

caesuraType (list)

An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)

An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

The first argument is the context.

The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Clef_engraver (page 413)

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.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly re-
   quested by the user.

forceBreak (boolean)
   Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and
ClefModifier (page 513).

Collision_engraver (page 413)
   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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
   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.

   forbidBreak (boolean)
      If set to #t, prevent a line break at this point, except if explicitly re-
      quested by the user.

   forceBreak (boolean)
      Set to #t when an event forcing a line break was heard.

   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): ClefModifier (page 513),
CueClef (page 521), and CueEndClef (page 524).
Custos_engraver (page 417)
  Engrave custodes.
  Properties (read)
    forbidBreak (boolean)
      If set to #t, prevent a line break at this point, except if explicitly re-
      quested by the user.
    forceBreak (boolean)
      Set to #t when an event forcing a line break was heard.
  This engraver creates the following layout object(s): Custos (page 527).

Dot_column_engraver (page 418)
  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): DotColumn (page 531).

Figured_bass_engraver (page 421)
  Make figured bass numbers.
  Music types accepted: bass-figure-event (page 49), and rest-event (page 55).
  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 exten-
      der lines.
    useBassFigureExtenders (boolean)
      Whether to use extender lines for repeated bass figures.
  This engraver creates the following layout object(s): BassFigure (page 491),
  BassFigureAlignment (page 492), BassFigureBracket (page 493),
  BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
  Position figured bass alignments over notes.
  This engraver creates the following layout object(s):
  BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
  Find potentially colliding scripts and put them into a FingeringColumn object; that
  will fix the collisions.
  This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
  Put fontSize into font-size grob property.
Properties (read)

`fontSize` (number)
The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
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.).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

`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): InstrumentName (page 559).

Key_engraver (page 429)
Engrave a key signature.

Music types accepted: `key-change-event` (page 52),

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.
forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-

quested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

keyAlterationOrder (list)
A list of pairs that defines in what order alterations should be printed.
The format of an entry is (step . alter), where step is a number from
0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
rationals for alterations in between, e.g., 1/2 for sharp.

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): KeyCancellation (page 563),
and KeySignature (page 565).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that nee d ledger lines.
This engraver creates the following layout object(s): LedgerLineSpanner
(page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all
have the same text and there are at least two only the first one is retained and the
others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn
object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

**Ottava_spanner_engraver** (page 439)
Create a text spanner when the ottavation property changes.
Music types accepted: ottava-event (page 54),
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): OttavaBracket (page 599).

**Output_property_engraver** (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

**Piano_pedal_align_engraver** (page 442)
Align piano pedal symbols and brackets.
Properties (read)

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

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

**Piano_pedal_engraver** (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),
Properties (read)

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

- *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): PianoPedalBracket
   (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and
   UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)
   Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
   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): RestCollision (page 612).

Script_row_engraver (page 445)
   Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
   Generate objects for computing spacing parameters.

Properties (read)
   createSpacing (boolean)
      Create StaffSpacing objects? Should be set for staves.

Properties (write)
   hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Signum_repetitionis_engraver (page 446)
   Create a SignumRepetitionis at the end of a \repeat volta section.
   Music types accepted: volta-repeat-end-event (page 59),
   This engraver creates the following layout object(s): SignumRepetitionis
   (page 618).

Skip_typesetting_engraver (page 446)
   Create a StaffEllipsis when skipTypesetting is used.

Properties (read)
   skipTypesetting (boolean)
      If true, no typesetting is done, speeding up the interpretation phase.
      Useful for debugging large scores.
This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),

Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),

This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),

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, \( \frac{4}{4} \) is a 4/4 time signature.

This engraver creates the following layout object(s): TimeSignature (page 655).

2.1.27 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 (page 390).
Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), MensuralLigature (page 585), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), Stem (page 634), StemStub (page 636), StemTremolo (page 637), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), and VoiceFollower (page 670).

This context sets the following properties:

- Set context property autoBeaming to #f.
- Set grob property length in Stem (page 634), to 5.
- Set grob property style in NoteHead (page 596), to 'petrucci.
- Set grob property style in Rest (page 611), to 'mensural.
- Set grob property thickness in Stem (page 634), to 1.7.

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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.

Music types accepted: arpeggio-event (page 49),

This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 49),

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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams.
Music types accepted: beam-event (page 49),
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): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.
Music types accepted: bend-after-event (page 49),
Properties (read)
  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
  currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)
  breathMarkType (symbol)
  The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).
Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514),
and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
    quested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner (page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-
event (page 49), and span-dynamic-event (page 56),
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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.
Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

Fingering_engraver (page 422)
Create fingering scripts.
Music types accepted: fingering-event (page 51),
This engraver creates the following layout object(s): Fingering (page 545).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Forbid_line_break_engraver (page 423)
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, except if explicitly requested by the user.

Glissando_engraver (page 424)
Engrave glissandi.
Music types accepted: glissando-event (page 51),
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): Glissando (page 552).

**Grace_auto_beam_engraver** (page 424)
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: beam-forbid-event (page 49),

Properties (read)

  autoBeaming (boolean)
  If set to true then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).

**Grace_beam_engraver** (page 425)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engravles beams when we are at grace points in time.

Music types accepted: beam-event (page 49),

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): Beam (page 495).

**Grace_engraver** (page 425)
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.

**Grob_pq_engraver** (page 426)
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.).
Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

  instrumentCueName (markup)
  The name to print if another instrument is to be taken.
  This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52).
This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

Mensural_ligature_engraver (page 435)
Handle Mensural_ligature_events by glueing special ligature heads together.
Music types accepted: ligature-event (page 52).
This engraver creates the following layout object(s): MensuralLigature (page 585).

Multi_measure_rest_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.
Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)

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

  internalBarNumber (integer)
  Contains the current bar number. This property is used for internal time-keeping, among others by the Accidental_engraver.

  measureStartNow (boolean)
  True at the beginning of a measure.

  restNumberThreshold (number)
  If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and MultiMeasureRestText (page 592).

New_fingering_engraver (page 437)
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): Fingering (page 545), Script (page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)
   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): VoiceFollower (page 670).

Note_heads_engraver (page 438)
   Generate note heads.
Music types accepted: note-event (page 54),
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): NoteHead (page 596).

Note_spacing_engraver (page 439)
   Generate NoteSpacing, an object linking horizontal lines for use in spacing.
This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
   Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
   Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.
Music types accepted: note-event (page 54), and part-combine-event (page 54),
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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)
  countPercentRepeats (boolean)
    If set, produce counters for percent repeats.
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55).
This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
Print the bracketed note head after a note head with trill.
This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).
Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

Slur_engraver (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Stem_engraver (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and triplet-span-event (page 58),
Properties (read)
currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.
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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

Text_ engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_ engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).

Tie_ engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and TieColumn (page 655).

Trill_spanner_ engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)
   currentCommandColumn (graphical (layout) object)
      Grob that is X-parent to all current breakable items (clef, key signature, etc.).
   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): TrillSpanner (page 661).
Tuplet_ engraver (page 455)

Catch tuplet events and generate appropriate bracket.

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): TupletBracket (page 662), and TupletNumber (page 664).

2.1.28 PianoStaff

Just like GrandStaff, but the staves are only removed together, never separately.

This context also accepts commands for the following context(s): GrandStaff (page 135).

This context creates the following layout object(s): Arpeggio (page 483), InstrumentName (page 559), SpanBar (page 626), SpanBarStub (page 628), StaffGrouper (page 630), SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), and VerticalAlignment (page 668).

This context sets the following properties:

- Set context property instrumentName to '().
- Set context property localAlterations to #f.
- Set context property localAlterations to '().
- Set context property localAlterations to '().
- Set context property shortInstrumentName to '().
- Set context property systemStartDelimiter to 'SystemStartBrace.
- Set context property systemStartDelimiter to 'SystemStartBracket.
- Set context property topLevelAlignment to #f.
- Set grob property extra-spacing-width in DynamicText (page 539), to #f.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Staff (page 287).

Context PianoStaff can contain ChoirStaff (page 66), ChordNames (page 95), Devnull (page 108), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

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

Instrument_name_ engraver (page 427)
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 items (clef, key signature, etc.).
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): InstrumentName (page 559).

Keep_alive_together_engraver (page 429)
   This engraver collects all Hara_kiri_group_spanners 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.

Output_property_engraver (page 439)
   Apply a procedure to any grob acknowledged.
   Music types accepted: apply-output-event (page 48),
Span_arpeggio_engraver (page 447)
   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): Arpeggio (page 483).

Span_bar_engraver (page 448)
   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): SpanBar (page 626).

Span_bar_stub_engraver (page 448)
   Make stubs for span bars in all contexts that the span bars cross.
   This engraver creates the following layout object(s): SpanBarStub (page 628).

System_start_delimiter_engraver (page 450)
   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 items (clef, key signature,
etc.).
      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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

Vertical_align_engraver (page 456)
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): StaffGrouper (page 630), and VerticalAlignment (page 668).

2.1.29 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 (page 287).

This context creates the following layout object(s): BarLine (page 486), BreathingSign (page 502), CaesuraScript (page 504), DotColumn (page 531), InstrumentName (page 559), LedgerLineSpanner (page 570), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), TimeSignature (page 655), and VerticalAxisGroup (page 668).

This context sets the following properties:

• Set context property createSpacing to #t.
• Set context property instrumentName to '().
• Set context property localAlterations to '().
• Set context property shortInstrumentName to '().
• Set context property squashedPosition to 0.
• Set grob property line-count in StaffSymbol (page 632), to 1.
• Set grob property neutral-direction in Beam (page 495), to 1.
• Set grob property neutral-direction in Stem (page 634), to 1.
• Set grob property staff-padding in VoltaBracket (page 671), to 3.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Voice (page 390).

Context RhythmicStaff can contain CueVoice (page 97), NullVoice (page 228), and Voice (page 390).

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

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).
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): VerticalAxisGroup
   (page 668).

Bar_engraver (page 404)
   Create bar lines for various commands, including '\bar'.
   If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.
   Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50),
   coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51),
   section-event (page 55), segno-mark-event (page 56), and volta-span-event
   (page 59),

Properties (read)
   caesuraType (list)
      An alist
         ((bar-line . bar-type)
         (breath . breath-type)
         (scripts . script-type...)
         (underlying-bar-line . bar-type))
      specifying which breath mark, bar line, and scripts to create at
      \caesura. All entries are optional.
      bar-line has higher priority than a measure bar line and underlying-
      bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of
   the same kind as caesuraType.
   The first argument is the context.
   The second argument is the value of caesuraType with an additional
   entry (articulations . symbol-list) identifying the articulations at-
   tached to the caesura in the music. If the transform function returns this
   second argument unmodified, it is as if no transform function were set;
   the function is free to return a different value. The transform function
   can remove articulations, but any added articulations are ignored.
   The third argument is a symbol-list identifying certain things the en-
   graver has observed. bar-line indicates that the engraver has observed
   a BarLine at the current moment.

doubleRepeatBarType (string)
   Bar line to insert where the end of one \repeat volta coincides with the
   start of another. The default is ':::'.

doubleRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with the end of one
   \repeat volta and the beginning of another. The default is ':|.S.|:'.
endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:\.’.

endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:\.S’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘:\.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘:\.S’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘:\.S.|:\’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ‘(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., ‘((start-repeat)) may be given as ‘(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘:\:\’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.\:\’.
underlyingRepeatBarType (string)
   Bar line to insert at points of repetition or departure where no bar line
   would normally appear, for example at the end of a system broken in
   mid measure where the next system begins with a segno. Where there is
   also a repeat bar line, the repeat bar line takes precedence and this value
   is appended to it as an annotation. The default is ‘||’.

whichBar (string)
   The current bar line type, or ’()’ if there is no bar line. Setting this ex-
   plicitly in user code is deprecated. Use \bar or related commands to set
   it.

Properties (write)

currentBarLine (graphical (layout) object)
   Set to the BarLine that Bar_engraver has created in the current
timestep.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly re-
   quested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
   Notate a short break in sound that does not shorten the previous note.
   Depending on the result of passing the value of caesuraType through
   caesuraTypeTransform, this engraver may create a BreathingSign with
   CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align
   them to a BarLine.
   If this engraver observes a BarLine, it calls caesuraTypeTransform again with the
   new information, and if necessary, recreates its grobs.
   Music types accepted: caesura-event (page 50),

Properties (read)

breathMarkDefinitions (list)
   The description of breath marks. This is used by the Breathing_sign_
   engraver. See scm/breath.scm for more information.

caesuraType (list)
   An alist
         ((bar-line . bar-type)
          (breath . breath-type)
          (scripts . script-type...)
          (underlying-bar-line . bar-type))
   specifying which breath mark, bar line, and scripts to create at
   \caesura. All entries are optional.
   bar-line has higher priority than a measure bar line and underlying-
   bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of
   the same kind as caesuraType.
   The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Dot_column_engraver (page 418)
   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): DotColumn (page 531).

Font_size_engraver (page 422)
   Put fontSize into font-size grob property.
   Properties (read)
      fontSize (number)
         The relative size of all grobs in a context.

Instrument_name_engraver (page 427)
   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 items (clef, key signature, etc.).
      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): InstrumentName (page 559).

Ledger_line_engraver (page 431)
   Create the spanner to draw ledger lines, and notices objects that need ledger lines.
   This engraver creates the following layout object(s): LedgerLineSpanner (page 570).
Output_property_engraver (page 439)
   Apply a procedure to any grob acknowledged.
   Music types accepted: apply-output-event (page 48),

Pitch_squash_engraver (page 443)
   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.

Separating_line_group_engraver (page 445)
   Generate objects for computing spacing parameters.
   Properties (read)
      createSpacing (boolean)
      Create StaffSpacing objects? Should be set for staves.
   Properties (write)
      hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.
   This engraver creates the following layout object(s): StaffSpacing (page 632).

Staff_highlight_engraver (page 449)
   Highlights music passages.
   Music types accepted: staff-highlight-event (page 56),
   Properties (read)
      currentCommandColumn (graphical (layout) object)
      Grob that is X-parent to all current breakable items (clef, key signature,
      etc.).
   This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
   Create the constellation of five (default) staff lines.
   Music types accepted: staff-span-event (page 57),
   This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
   Create a Section 3.1.146 [TimeSignature], page 655, whenever
   timeSignatureFraction changes.
   Music types accepted: time-signature-event (page 58),
   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): TimeSignature (page 655).
2.1.30 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.

An alias for Timing is established by the Timing_translator in whatever context it is initialized, and the timing variables are then copied from wherever Timing had been previously established. The alias at Score level provides a target for initializing Timing variables in layout definitions before any Timing_translator has been run.

This context also accepts commands for the following context(s): Timing (page 263).

This context creates the following layout object(s): BarNumber (page 489), BreakAlignGroup (page 500), BreakAlignment (page 501), CenteredBarNumber (page 506), CenteredBarNumberLineSpanner (page 507), CodaMark (page 515), ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), GraceSpacing (page 553), JumpScript (page 561), LeftEdge (page 571), MetronomeMark (page 585), NonMusicalPaperColumn (page 593), PaperColumn (page 600), Parentheses (page 601), RehearsalMark (page 607), SectionLabel (page 615), SegnoMark (page 616), SpacingSpanner (page 626), StaffGrouper (page 630), SystemStartBar (page 644), SystemStartBracket (page 645), SystemStartSquare (page 647), VerticalAlignment (page 668), VoltaBracket (page 671), and VoltaBracketSpanner (page 672).

This context sets the following properties:
- Set context property additionalPitchPrefix to "".
- Set context property aDueText to "a2".
- Set context property alterationGlyphs to #f.
- Set context property alternativeRestores to: ' (measurePosition measureLength lastChord)
- Set context property associatedVoiceType to 'Voice.
- Set context property autoAccidentals to:
  ' (Staff #<procedure 5575a375c520 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
- Set context property autoBeamCheck to default-auto-beam-check.
- Set context property autoBeaming to #t.
- Set context property autoCautionaries to '().
- Set context property barCheckSynchronize to #f.
- Set context property barNumberFormatter to robust-bar-number-function.
- Set context property barNumberVisibility to first-bar-number-invisible-and-no-parenthesized-bar-numbers.
- Set context property beamHalfMeasure to #t.
- Set context property breathMarkDefinitions to:
  ' ((caesura
      (text #<procedure musicglyph-markup (layout props glyph-name)>
       "scripts.caesura.straight")
   (chantdoublebar
   (extra-spacing-width -1.0 . 0.0)
   (stencil


• Set context property breathMarkType to 'comma.'
• Set context property caesuraType to: '((breath . caesura))'
• Set context property centerBarNumbers to #f.
• Set context property chordNameExceptions to:

```lilypond
#<procedure ly:breathing-sign::finalis (_)>)
(Y-offset . 0.0))
(chantfullbar
(extra-spacing-width -1.0 . 0.0)
(stencil
.
#<procedure ly:breathing-sign::divisio-maxima (_)>)
(Y-offset . 0.0))
(chanthalfbar
(extra-spacing-height
.
#<procedure item::extra-spacing-height-including-staff (grob)>)
(extra-spacing-width -1.0 . 0.0)
(stencil
.
#<procedure ly:breathing-sign::divisio-maior (_)>)
(Y-offset . 0.0))
(chantquarterbar
(extra-spacing-height
.
#<procedure item::extra-spacing-height-including-staff (grob)>)
(extra-spacing-width -1.0 . 0.0)
(stencil
.
#<procedure ly:breathing-sign::divisio-minima (_)>>)  
(comma (text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.rcomma"))
(curvedcaesura
(text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.caesura.curved")
(outsidecomma
(outside-staff-priority . 40)
(text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.rcomma")
(spacer
(text #<procedure null-markup (layout props)>)

(tickmark
(outside-staff-priority . 40)
(text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.tickmark")
(upbow (outside-staff-priority . 40)
(text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.upbow")
(varcomma
(text #<procedure musicglyph-markup (layout props glyph-name)>  
"scripts.rvarcomma")))
```
'((#<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)>
    "ø"))
((#<Pitch ees' > #<Pitch ges' > #<Pitch bes' >)
  #<procedure super-markup (layout props arg)>
  "ø")
((#<Pitch ees' > #<Pitch ges' > #<Pitch beses' >)
  #<procedure concat-markup (layout props args)>
  ((#<procedure line-markup (layout props args)>
    ((#<procedure fontsize-markup (layout props increment arg)>
      2
      "•")
    )
  ))
  ((#<procedure super-markup (layout props arg)>
    "7")
  ))
((#<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")
  ))
((#<Pitch g' >)
  #<procedure line-markup (layout props args)>
  ((#<procedure super-markup (layout props arg)>
    "5")
  ))
((#<Pitch g' > #<Pitch c'' >)
  #<procedure line-markup (layout props args)>
  ((#<procedure super-markup (layout props arg)>
    "5")
  ))

- Set context property chordNameFunction to ignatzek-chord-names.
- Set context property chordNameLowercaseMinor to #f.
- Set context property chordNameSeparator to:
  '((#<procedure hspace-markup (layout props amount)>
    0.5)
• Set context property chordNoteNamer to '().
• Set context property chordPrefixSpacer to 0.
• Set context property chordRootNamer to note-name->markup.
• Set context property clefGlyph to "clefs.G".
• Set context property clefPosition to -2.
• Set context property clefTranspositionFormatter to clef-transposition-markup.
• Set context property codaMarkFormatter to #<procedure 5575a20b1a00 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4 (number context)>.
• Set context property completionFactor to unity-if-multimeasure.
• Set context property crescendoSpanner to 'hairpin.
• Set context property cueClefTranspositionFormatter to clef-transposition-markup.
• Set context property dalSegnoTextFormatter to format-dal-segno-text.
• Set context property decrescendoSpanner to 'hairpin.
• Set context property doubleRepeatBarType to ":::":
• Set context property doubleRepeatSegnoBarType to "::|.|:":
• Set context property drumStyleTable to #<hash-table 5575a30e5760 29/61>.
• Set context property endRepeatBarType to "|.":
• Set context property endRepeatSegnoBarType to "|.|S".
• Set context property explicitClefVisibility to:
  #(#t #t #t)
• Set context property explicitCueClefVisibility to:
  #(#f #t #t)
• Set context property explicitKeySignatureVisibility to:
  #(#t #t #t)
• Set context property extendersOverRests to #t.
• Set context property extraNatural to #t.
• Set context property figuredBassAlterationDirection to -1.
• Set context property figuredBassFormatter to format-bass-figure.
• Set context property figuredBassLargeNumberAlignment to 0.
• Set context property figuredBassPlusDirection to -1.
• Set context property figuredBassPlusStrokedAlist to:
  '(((2 . "figbass.twoplus")
    (4 . "figbass.fourplus")
    (5 . "figbass.fiveplus")
    (6 . "figbass.sixstroked")
    (7 . "figbass.sevenstroked")
    (9 . "figbass.ninestroked"))
• Set context property fineBarType to "|.":
• Set context property fineSegnoBarType to "|.|S".
• Set context property fineStartRepeatSegnoBarType to "|.|S.|:":
• Set context property fineText to "Fine".
• Set context property fingeringOrientations to:
  '(up down)
• Set context property firstClef to #t.
• Set context property forbidBreakBetweenBarLines to #t.
• Set context 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 context property harmonicAccidentals to #t.
• Set context property highStringOne to #t.
• Set context property initialTimeSignatureVisibility to:
  #+#(#f #t #t)
• Set context property instrumentTransposition to #<Pitch c'>.
• Set context property keepAliveInterfaces to:
  '(
   bass-figure-interface
   chord-name-interface
   cluster-beacon-interface
   dynamic-interface
   fret-diagram-interface
   lyric-syllable-interface
   note-head-interface
   tab-note-head-interface
   lyric-interface
   percent-repeat-interface
   stanza-number-interface)
• Set context 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)
• Set context property `lyricMelismaAlignment` to `-1`.
• Set context property `majorSevenSymbol` to:
  `(line-markup
   (font-size-markup
    (-3
     (triangle-markup
      #f)))
   ))
• Set context property `measureBarType` to "|".
• Set context property `melismaBusyProperties` to:
  `(melismaBusy
   slurMelismaBusy
   tieMelismaBusy
   beamMelismaBusy
   completionBusy)
• Set context property `metronomeMarkFormatter` to `format-metronome-markup`.
• Set context property `middleCClefPosition` to `-6`.
• Set context property `middleCPosition` to `-6`.
• Set context property `minorChordModifier` to:
  `(simple-markup
   "m")
• Set context property `noChordSymbol` to:
  `(simple-markup
   "N.C.")
• Set context property `noteNameFunction` to `note-name-markup`.
• Set context property `noteNameSeparator` to "/".
• Set context property `noteToFretFunction` to `determine-frets`.
• Set context property `partCombineTextsOnNote` to `#t`.
• Set context property `pedalSostenutoStrings` to:
  `("Sost. Ped." "*Sost. Ped." "*")`
• Set context property `pedalSostenutoStyle` to `mixed`.
• Set context property `pedalSustainStrings` to:
  `("Ped." "*Ped." "*")`
• Set context property pedalSustainStyle to 'text.
• Set context property pedalUnaCordaStrings to:
  '("una corda" "tre corde")
• Set context property pedalUnaCordaStyle to 'text.
• Set context property predefinedDiagramTable to #f.
• Set context property printAccidentalNames to #t.
• Set context property printKeyCancellation to #t.
• Set context property printOctaveNames to #f.
• Set context property printPartCombineTexts to #t.
• Set context property quotedCueEventTypes to:
  ' (note-event
   rest-event
   tie-event
   beam-event
   tuplet-span-event
   tremolo-event)
• Set context property quotedEventTypes to:
  '(StreamEvent)
• Set context property rehearsalMarkFormatter to #<procedure 5575a20b18c0 at
  /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4
  (number context)).
• Set context property rehearsalMark to 1.
• Set context property repeatCountVisibility to all-repeat-counts-visible.
• Set context property restNumberThreshold to 1.
• Set context 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 . 0.4)
(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)
"ulongfermata")
(padding . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(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))
(outsidecomma
(avoid-slur . around)
(direction . 1)
(padding . 0.2)
(script-stencil feta "lcomma" . "rcomma")
(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 . 0.4)
  (avoid-slur . around)
  (script-priority . 4000)
(direction . 1))
(signumcongruentiae
 (script-stencil
  feta
  "dsignumcongruentiae"
  .
  "usignumcongruentiae")
  (padding . 0.2)
  (avoid-slur . outside)
  (direction . 1))
(slashturn
 (script-stencil feta "slashturn" . "slashturn")
  (padding . 0.2)
  (avoid-slur . inside)
  (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)
  (script-priority . -50)
  (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 . 0.4)
  (avoid-slur . around)
  (script-priority . 4000)
  (direction . 1))
(veryshortfermata
  (script-stencil
   feta
   "dveryshortfermata"
   "uveryshortfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (script-priority . 4000)
  (direction . 1))

• Set context property sectionBarType to "||".
• Set context property segnoBarType to "S".
• Set context property segnoMarkFormatter to format-segno-mark-considering-bar-lines.
• Set context property segnoStyle to 'mark.
• Set context property slashChordSeparator to:
  '(
  (simple-markup (layout props str)>
  "/
  )

• Set context property soloIIText to "Solo II".
• Set context property soloText to "Solo".
• Set context property startRepeatBarType to ".|:".
• Set context property startRepeatSegnoBarType to "S.|:".
• Set context property stringNumberOrientations to:
  'up down
• Set context property stringOneTopmost to #t.
• Set context property stringTunings to:
  '(
  #<Pitch e >
  #<Pitch b >
  #<Pitch g >
  #<Pitch d >
  #<Pitch a, >
  #<Pitch e, >)

• Set context property strokeFingerOrientations to:
  'right
• Set context property subdivideBeams to #f.
• Set context property suspendMelodyDecisions to #f.
• Set context property systemStartDelimiter to 'SystemStartBar.
• Set context property tablatureFormat to fret-number-tablature-format.
• Set context property tabStaffLineLayoutFunction to tablature-position-on-lines.
• Set context property tieWaitForNote to #f.
• Set context property timeSignatureFraction to:
  '4.4
• Set context property timeSignatureSettings to:
  '(((2 . 2) (beamExceptions (end (1/32 8 8 8 8))))
   ((3 . 2)
    (beamExceptions (end (1/32 8 8 8 8))))


Chapter 2: Translation 277

((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 4 4 4 4))))
((9 . 4) (beamExceptions (end (1/32 8 8 8 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 8 8))))
((5 . 8) (beatStructure 3 2))
((8 . 8) (beatStructure 3 3 2))

- Set context property timing to #t.
- Set context property topLevelAlignment to #t.
- Set context property underlyingRepeatBarType to "||".

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Staff (page 287).

Context Score can contain ChoirStaff (page 66), ChordNames (page 95), Devnull (page 108), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetruchiStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

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

- Bar_number_engraver (page 406)

A bar number may be created at any bar line, subject to the barNumberVisibility callback. By default, 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 2.2.135 [Staff_collecting_engraver], page 448. This engraver usually creates BarNumber grobs, but when centerBarNumbers is true, it makes CenteredBarNumber grobs instead.

Properties (read)

- alternativeNumber (non-negative, exact integer)
  When set, the index of the current \alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple volte.

- alternativeNumberingStyle (symbol)
  The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.
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.

centerBarNumbers (boolean)
Whether to center bar numbers in their measure instead of aligning them on the bar line.

currentBarNumber (integer)
Contains the current bar number. This property is incremented at every bar line.

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

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

measurePosition (moment)
How much of the current measure have we had. This can be set manually to create incomplete measures.
stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): BarNumber (page 489), and CenteredBarNumber (page 506).

Beam_collision_engraver (page 407)
Help beams avoid colliding with notes and clefs in other voices.

Break_align_engraver (page 410)
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): BreakAlignGroup (page 500), BreakAlignment (page 501), and LeftEdge (page 571).

Centered_bar_number_align_engraver (page 411)
Group measure-centered bar numbers in a CenteredBarNumberLineSpanner so they end up on the same vertical position.

Properties (read)

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

This engraver creates the following layout object(s): CenteredBarNumberLineSpanner (page 507).

Concurrent_hairpin_engraver (page 415)
Collect concurrent hairpins.

Footnote_engraver (page 423)
Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 548).

Grace_spacing_engraver (page 425)
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): GraceSpacing (page 553).

Jump_engraver (page 428)
This engraver creates instructions such as *D.C.* and *Fine*, placing them vertically outside the set of staves given in the stavesFound context property.

If Jump_engraver is added or moved to another context, Staff_collecting_engraver (page 448), also needs to be there so that marks appear at the intended Y location.

Music types accepted: ad-hoc-jump-event (page 48), dal-segno-event (page 50), and fine-event (page 51).

Properties (read)

  codaMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.
Chapter 2: Translation


codaMarkFormatter (procedure)
A procedure that creates a coda mark (which in conventional D.S. al
Coda form indicates the start of the alternative endings), taking as ar-
guments the mark sequence number and the context. It should return a
markup object.

dalSegnoTextFormatter (procedure)
Format a jump instruction such as D.S.
The first argument is the context.
The second argument is the number of times the instruction is per-
formed.
The third argument is a list of three markups: start-markup, end-
markup, and next-markup.
If start-markup is #f, the form is da capo; otherwise the form is dal segno
and start-markup is the sign at the start of the repeated section.
If end-markup is not #f, it is either the sign at the end of the main body
of the repeat, or it is a Fine instruction. When it is a Fine instruction,
next-markup is #f.
If next-markup is not #f, it is the mark to be jumped to after performing
the body of the repeat, e.g., Coda.

finalFineTextVisibility (boolean)
Whether \fine at the written end of the music should create a Fine
instruction.

fineText (markup)
The text to print at \fine.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set
during the first timestep, 0 up to the first segno, 1 from the first to the
second segno, 2 from the second to the third segno, etc.

segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the
start of a repeated section), taking as arguments the mark sequence
number and the context. It should return a markup object.

stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): JumpScript (page 561).

Mark_engraver (page 432)
This engraver creates rehearsal marks, segno and coda marks, and section labels.
Mark_engraver creates marks, formats them, and places them vertically outside the
set of staves given in the stavesFound context property.

If Mark_engraver is added or moved to another context, Staff_collecting-
engraver (page 448), also needs to be there so that marks appear at the intended
Y location.

By default, Mark_engravers in multiple contexts create a common sequence of marks
chosen by the Score-level Mark_tracking_translator (page 433). If independent
sequences are desired, multiple Mark_tracking_translators must be used.
Properties (read)

codaMarkFormatter (procedure)
A procedure that creates a coda mark (which in conventional *D.S. al Coda* form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.

currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

rehearsalMarkFormatter (procedure)
A procedure taking as arguments the context and the sequence number of the rehearsal mark. It should return the formatted mark as a markup object.

segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.

stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): CodaMark (page 515), RehearsalMark (page 607), SectionLabel (page 615), and SegnoMark (page 616).

Mark_tracking_translator (page 433)
This translator chooses which marks Mark_engraver should engrave.

Music types accepted: ad-hoc-mark-event (page 48), coda-mark-event (page 50), rehearsal-mark-event (page 55), section-label-event (page 55), and segno-mark-event (page 56),

Properties (read)

codaMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

rehearsalMark (integer)
The next rehearsal mark to print.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

Properties (write)

codaMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.
currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

rehearsalMark (integer)
The next rehearsal mark to print.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

Metronome_mark_engraver (page 435)
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 2.2.135 [Staff_collecting_engraver], page 448.

Music types accepted: tempo-change-event (page 58),

Properties (read)

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

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): MetronomeMark (page 585).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Paper_column_engraver (page 440)
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: break-event (page 49), and label-event (page 52),
Properties (read)

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

Properties (write)

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

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, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): NonMusicalPaperColumn (page 593), and PaperColumn (page 600).

Parenthesis_engraver (page 440)
Parenthesize objects whose parenthesize property is #t.
This engraver creates the following layout object(s): Parentheses (page 601).

Repeat_acknowledge_engraver (page 443)
This translator adds entries to repeatCommands for events generated by \\repeat volta.

Music types accepted: volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59),

Properties (write)

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

Show_control_points_engraver (page 446)
Create grobs to visualize control points of Bézier curves (ties and slurs) for ease of tweaking.
This engraver creates the following layout object(s): ControlPoint (page 519), and ControlPolygon (page 520).
Spacing_ engraver (page 447)
Make a SpacingSpanner and do bookkeeping of shortest starting and playing notes.
Music types accepted: spacing-section-event (page 56),
Properties (read)
  currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): SpacingSpanner (page 626).

Spanner_tracking_ engraver (page 448)
Helper for creating spanners attached to other spanners. If a spanner has the sticky-grob-interface, the engraver tracks the spanner contained in its sticky-host object. When the host ends, the sticky spanner attached to it has its end announced too.

Staff_collecting_ engraver (page 448)
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.

Stanza_number_align_ engraver (page 449)
This engraver ensures that stanza numbers are neatly aligned.

System_start_delimiter_ engraver (page 450)
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 items (clef, key signature, etc.).
  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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).
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.

Music types accepted: alternative-event (page 48), bar-event (page 49), and fine-event (page 51).

Properties (read)

alternativeNumberingStyle (symbol)
The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

baseMoment (moment)
Smallest unit of time that will stand on its own as a subdivided section.

currentBarNumber (integer)
Contains the current bar number. This property is incremented at every bar line.

internalBarNumber (integer)
Contains the current bar number. This property is used for internal time-keeping, 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)

alternativeNumber (non-negative, exact integer)
When set, the index of the current alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple voltes.

baseMoment (moment)
Smallest unit of time that will stand on its own as a subdivided section.

currentBarNumber (integer)
Contains the current bar number. This property is incremented at every bar line.

internalBarNumber (integer)
Contains the current bar number. This property is used for internal time-keeping, 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.
measureStartNow (boolean)
True at the beginning of a measure.

timeSignatureFraction (fraction, as pair)
A pair of numbers, signifying the time signature. For example, '4 . 4)
is a 4/4 time signature.

Tweak_engraver (page 456)
Read the tweaks property from the originating event, and set properties.

Vertical_align_engraver (page 456)
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): StaffGrouper (page 630), and VerticalAlignment (page 668).

Volta_engraver (page 456)
Make volta brackets.
Music types accepted: dal-segno-event (page 50), fine-event (page 51), and volta-span-event (page 59),
Properties (read)
currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).
repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, 'command args . . . ), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as 'command.
   end-repeat
   End a repeated section.
   start-repeat
   Start a repeated section.
   volta text
   If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.
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): VoltaBracket (page 671), and VoltaBracketSpanner (page 672).
2.1.31 Staff

Handles clefs, bar lines, keys, accidentals. It can contain Voice contexts.

This context creates the following layout object(s): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), BreathingSign (page 502), CaesuraScript (page 504), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565), LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642), TimeSignature (page 655), UnaCordaPedal (page 665), UnaCordaPedallineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:

- Set context property createSpacing to #t.
- Set context property ignoreFiguredBassRest to #f.
- Set context property instrumentName to '.
- Set context property localAlterations to '('.
- Set context property ottavationMarkups to:

`'((4 . "29")
  (3 . "22")
  (2 . "15")
  (1 . "8")
  (-1 . "8")
  (-2 . "15")
  (-3 . "22")
  (-4 . "29"))`

- Set context property shortInstrumentName to '('.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type Voice (page 390).

Context Staff can contain CueVoice (page 97), NullVoice (page 228), and Voice (page 390).

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

Accidental_engraver (page 400)

Make accidentals. Catch note heads, ties and notices key-change events. This engraver usually lives at Staff level, but reads the settings for Accidentals 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.

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 bar number. This property is used for internal time-keeping, 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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), and AccidentalSuggestion (page 478).

Alteration_glyph_engraver (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context's alterationGlyphs property, when defined.

Properties (read)
alterationGlyphs (list)
A list mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)
Create bar lines for various commands, including \bar. If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)
caesuraType (list)
An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))
specifying which breath mark, bar line, and scripts to create at
\caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-
bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of
the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional
entry (articulations . symbol-list) identifying the articulations at-
tached to the caesura in the music. If the transform function returns this
second argument unmodified, it is as if no transform function were set;
the function is free to return a different value. The transform function
can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the en-
graver has observed. bar-line indicates that the engraver has observed
a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the
start of another. The default is ‘:\dots:’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one
\repeat volta and the beginning of another. The default is ‘:|.S.|:’.

doubleRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

de\repeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a
\repeat volta. The default is ‘:|.S’.

doubleRepeatBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. Where there is also a repeat bar line, the
repeat bar line takes precedence and this value is appended to it as an
annotation. The default is ‘|.’.

doubleRepeatBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The de-
fault is ‘|.S’.

doubleRepeatBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the
start of a \repeat volta. The default is ‘|:.S.:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar
line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each ele-
ment is a list, ‘(command args...)’, but a command with no arguments
may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
The current bar line type, or ‘() if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
Notate a short break in sound that does not shorten the previous note.
Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with
CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.

If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),

Properties (read)

breathMarkDefinitions (list)
   The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

caesuraType (list)
   An alist
   
   ((bar-line . bar-type)
    (breath . breath-type)
    (scripts . script-type...)
    (underlying-bar-line . bar-type))

   specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.
   bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
   The first argument is the context.
   The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
   The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Clef_engraver (page 413)
   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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.
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): ClefModifier (page 513),
CueClef (page 521), and CueEndClef (page 524).

Dot_column_engraver (page 418)
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): DotColumn (page 531).

Figured_bass_engraver (page 421)
Make figured bass numbers.
Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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 exten-
der lines.

useBassFigureExtenders (boolean)
Whether to use extender lines for repeated bass figures.

This engraver creates the following layout object(s): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.
This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that
will fix the collisions.
This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.
Grob\_pq\_engraver (page 426)

Administrate when certain grobs (e.g., note heads) stop playing.

Properties (read)

\texttt{busyGrobs} (list)

A queue of \texttt{(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)

\texttt{busyGrobs} (list)

A queue of \texttt{(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.).

Instrument\_name\_engraver (page 427)

Create a system start text for instrument or vocal names.

Properties (read)

\texttt{currentCommandColumn} (graphical (layout) object)

Grob that is X-parent to all current breakable items (clef, key signature, etc.).

\texttt{instrumentName} (markup)

The name to print left of a staff. The \texttt{instrumentName} property labels the staff in the first system, and the \texttt{shortInstrumentName} property labels following lines.

\texttt{shortInstrumentName} (markup)

See \texttt{instrumentName}.

\texttt{shortVocalName} (markup)

Name of a vocal line, short version.

\texttt{vocalName} (markup)

Name of a vocal line.

This engraver creates the following layout object(s): \texttt{InstrumentName} (page 559).

Key\_engraver (page 429)

Engrave a key signature.

Music types accepted: key-change-event (page 52),

Properties (read)

\texttt{createKeyOnClefChange} (boolean)

Print a key signature whenever the clef is changed.

\texttt{explicitKeySignatureVisibility} (vector)

‘\texttt{break-visibility}’ function for explicit key changes. ‘\texttt{\override}’ of the \texttt{break-visibility} property will set the visibility for normal (i.e., at the start of the line) key signatures.

\texttt{extraNatural} (boolean)

Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

\texttt{forbidBreak} (boolean)

If set to \texttt{#t}, prevent a line break at this point, except if explicitly requested by the user.
forceBreak (boolean)
   Set to #t when an event forcing a line break was heard.

keyAlterationOrder (list)
   A list of pairs that defines in what order alterations should be printed.
   The format of an entry is (step . alter), where step is a number from 0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
   rationals for alterations in between, e.g., 1/2 for sharp.

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): KeyCancellation (page 563),
and KeySignature (page 565).

Ledger_line_engraver (page 431)
   Create the spanner to draw ledger lines, and notices objects that need ledger lines.
   This engraver creates the following layout object(s): LedgerLineSpanner
   (page 570).

Merge_mmrest_numbers_engraver (page 435)
   Engraver to merge multi-measure rest numbers in multiple voices.
   This works by gathering all multi-measure rest numbers at a time step. If they all
   have the same text and there are at least two only the first one is retained and the
   others are hidden.

Non_musical_script_column_engraver (page 437)
   Find potentially colliding non-musical scripts and put them into a ScriptColumn
   object; that will fix the collisions.
   This engraver creates the following layout object(s): ScriptColumn (page 614).
Ottava_spanner_engraver (page 439)
Create a text spanner when the ottavation property changes.
Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.

Properties (read)

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

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),

Properties (read)

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

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): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)  
Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)  
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): RestCollision (page 612).

Script_row_engraver (page 445)  
Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)  
Generate objects for computing spacing parameters.

Properties (read)

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

Properties (write)

hasStaffSpacing (boolean)  
True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)  
Create a StaffEllipsis when skipTypesetting is used.

Properties (read)

skipTypesetting (boolean)  
If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)  
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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),
Properties (read)
currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),
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): TimeSignature (page 655).

2.1.32 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): Arpeggio (page 483), InstrumentName
(page 559), SpanBar (page 626), SpanBarStub (page 628), StaffGrouper (page 630),
SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646),
SystemStartSquare (page 647), and VerticalAlignment (page 668).

This context sets the following properties:
• Set context property instrumentName to '().
• Set context property localAlterations to #f.
• Set context property localAlterations to '().
• Set context property shortInstrumentName to '().
• Set context property systemStartDelimiter to 'SystemStartBracket.
• Set context property topLevelAlignment to #f.
• Set grob property extra-spacing-width in DynamicText (page 539), to #f.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit
context of type Staff (page 287).
Context StaffGroup can contain ChoirStaff (page 66), ChordNames (page 95), Devnull (page 108), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

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

Instrument_name_engraver (page 427)
  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 items (clef, key signature, etc.).
    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): InstrumentName (page 559).

Output_property_engraver (page 439)
  Apply a procedure to any grob acknowledged.
  Music types accepted: apply-output-event (page 48),

Span_arpeggio_engraver (page 447)
  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): Arpeggio (page 483).

Span_bar_engraver (page 448)
  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): SpanBar (page 626).

Span_bar_stub_engraver (page 448)
  Make stubs for span bars in all contexts that the span bars cross.
  This engraver creates the following layout object(s): SpanBarStub (page 628).

System_start_delimiter_engraver (page 450)
  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 items (clef, key signature, etc.).

  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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

  Vertical_align_engraver (page 456)
    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): StaffGrouper (page 630), and VerticalAlignment (page 668).

2.1.33 StandaloneRhythmScore

A Score-level context for use by \markup \rhythm.

This context also accepts commands for the following context(s): Score (page 263), and Timing (page 263).

This context creates the following layout object(s): BarNumber (page 489), BreakAlignGroup (page 500), BreakAlignment (page 501), CenteredBarNumberLineSpanner (page 507), CodaMark (page 515), ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), GraceSpacing (page 553), JumpScript (page 561), LeftEdge (page 571), MetronomeMark (page 585), NonMusicalPaperColumn (page 593), PaperColumn (page 600), Parentheses (page 601), RehearsalMark (page 607), SectionLabel (page 615), SegnoMark (page 616), SpacingSpanner (page 626), StaffGrouper (page 630), SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), VerticalAlignment (page 668), VoltaBracket (page 671), and VoltaBracketSpanner (page 672).

This context sets the following properties:

  • Set context property additionalPitchPrefix to "".
  • Set context property aDueText to "a2".
  • Set context property alterationGlyphs to #f.
  • Set context property alternativeRestores to:
    ' (measurePosition measureLength lastChord)
  • Set context property associatedVoiceType to 'Voice.
• Set context property autoAccidentals to:
  `'((Staff #<procedure 5575a375c520 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
  • Set context property autoBeamCheck to default-auto-beam-check.
• Set context property autoBeaming to #t.
• Set context property autoCautionaries to '().
• Set context property barCheckSynchronize to #f.
• Set context property barNumberFormatter to robust-bar-number-function.
• Set context property barNumberVisibility to first-bar-number-invisible-and-no-parenthesized-bar-numbers.
• Set context property beamHalfMeasure to #t.
• Set context property breathMarkDefinitions to:
  `'((caesura
    (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.caesura.straight")
  (chantdoublebar
    (extra-spacing-width -1.0 0.0)
    (stencil
      .
      #<procedure ly:breathing-sign::finalis (_)>)
    (Y-offset . 0.0))
  (chantfullbar
    (extra-spacing-width -1.0 0.0)
    (stencil
      .
      #<procedure ly:breathing-sign::divisio-maxima (_)>)
    (Y-offset . 0.0))
  (chanthalfbar
    (extra-spacing-height
      .
      #<procedure item::extra-spacing-height-including-staff (grob)>)
    (extra-spacing-width -1.0 0.0)
    (stencil
      .
      #<procedure ly:breathing-sign::divisio-maior (_)>)
    (Y-offset . 0.0))
  (chantquarterbar
    (extra-spacing-height
      .
      #<procedure item::extra-spacing-height-including-staff (grob)>)
    (extra-spacing-width -1.0 0.0)
    (stencil
      .
      #<procedure ly:breathing-sign::divisio-minima (_)>))
  (comma (text #<procedure musicglyph-markup (layout props glyph-name)>
    "scripts.rcomma")
  (curvedcaesura
    (text #<procedure musicglyph-markup (layout props glyph-name)>
      "scripts.caesura.curved")
  (outsidecomma
    (outside-staff-priority . 40)
Set context property breathMarkType to 'comma.

Set context property caesuraType to:
'((breath . caesura))

Set context property centerBarNumbers to #f.

Set context property chordNameExceptions to:
'(((#<Pitch e' > #<Pitch gis' >)
#<procedure line-markup (layout props args)>  
("+"))
((#<Pitch ees' > #<Pitch ges' >)
#<procedure line-markup (layout props args)>  
((#<procedure fontsize-markup (layout props increment arg)>  
2
" ● "))))
((#<Pitch ees' > #<Pitch ges' > #<Pitch bes' >)
#<procedure line-markup (layout props args)>  
((#<procedure super-markup (layout props arg)>  
"ø"))
((#<Pitch ees' > #<Pitch ges' > #<Pitch beses' >)
#<procedure concat-markup (layout props args)>  
((#<procedure fontsize-markup (layout props increment arg)>  
2
" "))))
((#<procedure super-markup (layout props arg)>  
"7")))))
((#<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'' >
• Set context property chordNameFunction to ignatzek-chord-names.
• Set context property chordNameLowercaseMinor to #f.
• Set context property chordNameSeparator to: '('#<procedure hspace-markup (layout props amount)> 0.5)
• Set context property chordNoteNamer to '().
• Set context property chordPrefixSpacer to 0.
• Set context property chordRootNamer to note-name->markup.
• Set context property clefGlyph to "clefs.G".
• Set context property clefPosition to -2.
• Set context property clefTranspositionFormatter to clef-transposition-markup.
• Set context property codaMarkFormatter to #<procedure 5575a20b1a00 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4 (number context)>.
• Set context property completionFactor to unity-if-multimeasure.
• Set context property crescendoSpanner to 'hairpin.
• Set context property cueClefTranspositionFormatter to clef-transposition-markup.
• Set context property dalSegnoTextFormatter to format-dal-segno-text.
• Set context property decrescendoSpanner to 'hairpin.
• Set context property doubleRepeatBarType to ":...:"
• Set context property doubleRepeatSegnoBarType to ":|.S.|:".
• Set context property drumStyleTable to #<hash-table 5575a306760 29/61>.
• Set context property endRepeatBarType to ":|.".
• Set context property endRepeatSegnoBarType to ":|.S".
• Set context property explicitClefVisibility to: #(#t #t #t)
• Set context property explicitCueClefVisibility to: #(#t #t #t)
• Set context property explicitKeySignatureVisibility to:
  #(#t #t #t)
• Set context property extendersOverRests to #t.
• Set context property extraNatural to #t.
• Set context property figuredBassAlterationDirection to -1.
• Set context property figuredBassFormatter to format-bass-figure.
• Set context property figuredBassLargeNumberAlignment to 0.
• Set context property figuredBassPlusDirection to -1.
• Set context property figuredBassPlusStrokedAlist to:
  '((2 . "figbass.twoplus")
   (4 . "figbass.fourplus")
   (5 . "figbass.fiveplus")
   (6 . "figbass.sixstroked")
   (7 . "figbass.sevenstroked")
   (9 . "figbass.ninestroked"))
• Set context property fineBarType to "|.".
• Set context property fineSegnoBarType to "|.S".
• Set context property fineStartRepeatSegnoBarType to "|.S.:".
• Set context property fineText to "Fine".
• Set context property fingeringOrientations to:
  '((up down)
   (down up))
• Set context property firstClef to #t.
• Set context property forbidBreakBetweenBarLines to #t.
• Set context 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 context property harmonicAccidentals to #t.
• Set context property highStringOne to #t.
• Set context property initialTimeSignatureVisibility to:
  #(#f #t #t)
• Set context property instrumentTransposition to <Pitch c'>.
• Set context property keepAliveInterfaces to:
  '(bass-figure-interface
   chord-name-interface
   cluster-beacon-interface
   dynamic-interface
   fret-diagram-interface
lyric-syllable-interface
note-head-interface
tab-note-head-interface
lyric-interface
percent-repeat-interface
stanza-number-interface)

• Set context 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 context property lyricMelismaAlignment to -1.

• Set context property majorSevenSymbol to:
'(#<procedure line-markup (layout props args)>
 ((#<procedure fontsize-markup (layout props increment arg)>
   -3
   (#<procedure triangle-markup (layout props filled)>
    #f)))))

• Set context property measureBarType to "|".

• Set context property melismaBusyProperties to:
'((melismaBusy
  slurMelismaBusy
  tieMelismaBusy
  beamMelismaBusy
  completionBusy)

• Set context property metronomeMarkFormatter to format-metronome-markup.
• Set context property middleCClefPosition to -6.
• Set context property middleCPosition to -6.
• Set context property minorChordModifier to:
  '(%<procedure simple-markup (layout props str)>
   "m")
• Set context property noChordSymbol to:
  '(%<procedure simple-markup (layout props str)>
   "N.C.")
• Set context property noteNameFunction to note-name-markup.
• Set context property noteNameSeparator to "/".
• Set context property noteToFretFunction to determine-frets.
• Set context property partCombineTextsOnNote to #t.
• Set context property pedalSostenutoStrings to:
• Set context property pedalSostenutoStyle to 'mixed.
• Set context property pedalSustainStrings to:
  '("Ped." "*Ped." "*")
• Set context property pedalSustainStyle to 'text.
• Set context property pedalUnaCordaStrings to:
  '("una corda" "tre corde")
• Set context property pedalUnaCordaStyle to 'text.
• Set context property predefinedDiagramTable to #f.
• Set context property printAccidentalNames to #t.
• Set context property printKeyCancellation to #t.
• Set context property printOctaveNames to #f.
• Set context property printPartCombineTexts to #t.
• Set context property quotedCueEventTypes to:
  '(%note-event
    %rest-event
    %tie-event
    %beam-event
    %tuplet-span-event
    %tremolo-event)
• Set context property quotedEventTypes to:
  '(%StreamEvent)
• Set context property rehearsalMarkFormatter to #<procedure 5575a20b18c0 at
  /build/out/share/lilypond/current/scm/lily/translation-functions.scm:224:4
  (number context)>.
• Set context property rehearsalMark to 1.
• Set context property repeatCountVisibility to all-repeat-counts-visible.
• Set context property restNumberThreshold to 1.
• Set context property scriptDefinitions to:
  '(%accent
    (avoid-slur . around)
    (padding . 0.2)
(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 0.4)
  (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))

(haydnturn
  (script-stencil feta "haydnturn" "haydnturn")
  (padding 0.2)
  (avoid-slur inside)
  (direction 1))

(henzelongfermata
  (script-stencil feta "dhenzelongfermata" "uhenzelongfermata")
  (padding 0.4)
  (avoid-slur around)
  (script-priority 4000)
  (direction 1))

(henzeshortfermata
  (script-stencil feta "dhenzeshortfermata" "uhenzeshortfermata")
  (padding 0.4)
  (avoid-slur around)
  (script-priority 4000)
  (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 . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(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))

(outsidecomma
(avoid-slur . around)
(direction . 1)
(padding . 0.2)
(script-stencil feta "lcomma" . "rcomma")

(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))
(semicolon
(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 . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(direction . 1))
(signumcongruentiae
(script-stencil
 feta
 "dsignumcongruentiae"
 .
 "usignumcongruentiae")
(padding . 0.2)
(avoid-slur . outside)
(direction . 1))
(slashturn
(script-stencil feta "slashturn" . "slashturn")
(padding . 0.2)
(avoid-slur . inside)
(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")
(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)
(script-priority . -50)
(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 "varcodad. "varcodad")
(padding . 0.2)
(avoid-slur . outside)
(direction . 1))
(varcomma
(script-stencil feta "lvarcommad. "rvarcomma")
(quantize-position . #t)
(padding . 0.2)
(avoid-slur . ignore)
(direction . 1))
(verylongfermata
(script-stencil
 feta
 "dverylongfermata"
 .
 "uverylongfermata")
(padding . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(direction . 1))
(veryshortfermata
(script-stencil
 feta
 "dveryshortfermata"
 .
 "uveryshortfermata")
(padding . 0.4)
(avoid-slur . around)
(script-priority . 4000)
(direction . 1)))

- Set context property sectionBarType to "||".
- Set context property segnoBarType to "S".
- Set context property segnoMarkFormatter to format-segno-mark-considering-bar-lines.
- Set context property segnoStyle to 'mark.
- Set context property slashChordSeparator to:
  '('#<procedure simple-markup (layout props str)> 
    "/"')
- Set context property soloIIText to "Solo II".
- Set context property soloText to "Solo".
- Set context property startRepeatBarType to ".|:".
- Set context property startRepeatSegnoBarType to "S.|:".
- Set context property stringNumberOrientations to:
  '(up down)
- Set context property stringOneTopmost to #t.
- Set context property stringTunings to:
  '('#<Pitch e'>
    #<Pitch b'>
    #<Pitch g'>
• Set context property `strokeFinger Orientations` to: 
  `(right)`
• Set context property `subdivideBeams` to `#f`.
• Set context property `suspendMelodyDecisions` to `#f`.
• Set context property `systemStartDelimiter` to `SystemStartBar`.
• Set context property `tablatureFormat` to `fret-number-tablature-format`.
• Set context property `tabStaffLineLayoutFunction` to `tablature-position-on-lines`.
• Set context property `tieWaitForNote` to `#f`.
• Set context property `timeSignatureFraction` to: 
  `((4 . 4)`
• Set context property `timeSignatureSettings` to:
  `'(((2 . 2) (beamExceptions (end (1/32 8 8 8 8))))
   ((3 . 2) 
   (beamExceptions (end (1/32 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) 
   (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))))
   ((9 . 4) 
   (beamExceptions (end (1/32 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 context property `timing` to `#f`.
• Set context property `topLevelAlignment` to `#t`.
• Set context property `underlyingRepeatBarType` to `"||"`.
• Set grob property `common-shortest-duration` in `SpacingSpanner` (page 626), to `#<Mom 1/10>`.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `StandaloneRhythmStaff` (page 325).

Context `StandaloneRhythmScore` can contain `ChoirStaff` (page 66), `ChordNames` (page 95), `Devnull` (page 108), `DrumStaff` (page 108), `Dynamics` (page 126), `FiguredBass` (page 132), `FretBoards` (page 133), `GrandStaff` (page 135), `GregorianTranscriptionLyrics` (page 137), `GregorianTranscriptionStaff` (page 140), `KievanStaff` (page 176), `Lyrics` (page 199), `MensuralStaff` (page 202), `NoteNames` (page 226), `OneStaff` (page 230),
Bar number may be created at any bar line, subject to the barNumberVisibility callback. By default, 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 2.2.135 [Staff_collecting_engraver], page 448. This engraver usually creates BarNumber grobs, but when centerBarNumbers is true, it makes CenteredBarNumber grobs instead.

Properties (read)

alternativeNumber (non-negative, exact integer)
When set, the index of the current alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple voltes.

alternativeNumberingStyle (symbol)
The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

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-in 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 \textit{n} \textit{m})

If bar numbers 1, 4, 7, etc., should be enabled, \textit{n} (the modulo) must be set to 3 and \textit{m} (the division remainder) to 1.

centerBarNumbers \text{(boolean)}
Whether to center bar numbers in their measure instead of aligning them on the bar line.

currentBarNumber \text{(integer)}
Contains the current bar number. This property is incremented at every bar line.

currentCommandColumn \text{(graphical (layout) object)}
Grob that is X-parent to all current breakable items (clef, key signature, etc.).

forbidBreak \text{(boolean)}
If set to \#t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak \text{(boolean)}
Set to \#t when an event forcing a line break was heard.

measurePosition \text{(moment)}
How much of the current measure have we had. This can be set manually to create incomplete measures.

stavesFound \text{(list of grobs)}
A list of all staff-symbols found.

This engraver creates the following layout object(s): BarNumber (page 489), and CenteredBarNumber (page 506).

Beam_collision_engraver (page 407)
Help beams avoid colliding with notes and clefs in other voices.

Break_align_engraver (page 410)
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): BreakAlignGroup (page 500), BreakAlignment (page 501), and LeftEdge (page 571).

Centered_bar_number_align_engraver (page 411)
Group measure-centered bar numbers in a CenteredBarNumberLineSpanner so they end up on the same vertical position.

Properties (read)

\begin{itemize}
  \item \text{currentCommandColumn} \text{(graphical (layout) object)}
    \begin{itemize}
      \item Grob that is X-parent to all current breakable items (clef, key signature, etc.).
    \end{itemize}
\end{itemize}

This engraver creates the following layout object(s):
CenteredBarNumberLineSpanner (page 507).

Concurrent_hairpin_engraver (page 415)
Collect concurrent hairpins.

Footnote_engraver (page 423)
Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 548).
Grace_spacing_ engraver (page 425)

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): GraceSpacing (page 553).

Jump_ engraver (page 428)

This engraver creates instructions such as D.C. and Fine, placing them vertically outside the set of staves given in the stavesFound context property.

If Jump_ engraver is added or moved to another context, Staff_collecting_ engraver (page 448), also needs to be there so that marks appear at the intended Y location.

Music types accepted: ad-hoc-jump-event (page 48), dal-segno-event (page 50), and fine-event (page 51).

Properties (read)

- codaMarkCount (non-negative, exact integer)
  Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

- codaMarkFormatter (procedure)
  A procedure that creates a coda mark (which in conventional D.S. al Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.

- dalSegnoTextFormatter (procedure)
  Format a jump instruction such as D.S. The first argument is the context. The second argument is the number of times the instruction is performed. The third argument is a list of three markups: start-markup, end-markup, and next-markup.
  If start-markup is #f, the form is da capo; otherwise the form is dal segno and start-markup is the sign at the start of the repeated section.
  If end-markup is not #f, it is either the sign at the end of the main body of the repeat, or it is a Fine instruction. When it is a Fine instruction, next-markup is #f.
  If next-markup is not #f, it is the mark to be jumped to after performing the body of the repeat, e.g., Coda.

- finalFineTextVisibility (boolean)
  Whether \fine at the written end of the music should create a Fine instruction.

- fineText (markup)
  The text to print at \fine.
segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set
during the first timestep, 0 up to the first segno, 1 from the first to the
second segno, 2 from the second to the third segno, etc.

segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the
start of a repeated section), taking as arguments the mark sequence
number and the context. It should return a markup object.

stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): JumpScript (page 561).

Mark_engraver (page 432)
This engraver creates rehearsal marks, segno and coda marks, and section labels.
Mark_engraver creates marks, formats them, and places them vertically outside the
set of staves given in the stavesFound context property.
If Mark_engraver is added or moved to another context, Staff_collecting_
engraver (page 448), also needs to be there so that marks appear at the intended
Y location.
By default, Mark_engravers in multiple contexts create a common sequence of marks
chosen by the Score-level Mark_tracking_translator (page 433). If independent
sequences are desired, multiple Mark_tracking_translators must be used.
Properties (read)

codaMarkFormatter (procedure)
A procedure that creates a coda mark (which in conventional D.S. al
Coda form indicates the start of the alternative endings), taking as arg-
ments the mark sequence number and the context. It should return a
markup object.
currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_
translator for engraving by Mark_engraver.
currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_
translator for engraving by Mark_engraver.
rehearsalMarkFormatter (procedure)
A procedure taking as arguments the context and the sequence number
of the rehearsal mark. It should return the formatted mark as a markup
object.
segnoMarkFormatter (procedure)
A procedure that creates a segno (which conventionally indicates the
start of a repeated section), taking as arguments the mark sequence
number and the context. It should return a markup object.
stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): CodaMark (page 515),
RehearsalMark (page 607), SectionLabel (page 615), and SegnoMark (page 616).
Mark_tracking_translator (page 433)
This translator chooses which marks Mark_engraver should engrave.
Music types accepted: ad-hoc-mark-event (page 48), coda-mark-event (page 50),
rehearsal-mark-event (page 55), section-label-event (page 55), and
segno-mark-event (page 56),
Properties (read)
  codaMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a coda mark appears: not
    set during the first timestep, 0 up to the first coda mark, 1 from the first
    to the second, 2 from the second to the third, etc.
  rehearsalMark (integer)
    The next rehearsal mark to print.
  segnoMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a segno appears: not set
    during the first timestep, 0 up to the first segno, 1 from the first to the
    second segno, 2 from the second to the third segno, etc.
Properties (write)
  codaMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a coda mark appears: not
    set during the first timestep, 0 up to the first coda mark, 1 from the first
    to the second, 2 from the second to the third, etc.
  currentPerformanceMarkEvent (stream event)
    The coda, section, or segno mark event selected by Mark_tracking_  
    translator for engraving by Mark_engraver.
  currentRehearsalMarkEvent (stream event)
    The ad-hoc or rehearsal mark event selected by Mark_tracking_  
    translator for engraving by Mark_engraver.
  rehearsalMark (integer)
    The next rehearsal mark to print.
  segnoMarkCount (non-negative, exact integer)
    Updated at the end of each timestep in which a segno appears: not set
    during the first timestep, 0 up to the first segno, 1 from the first to the
    second segno, 2 from the second to the third segno, etc.
Metronome_mark_engraver (page 435)
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 2.2.135
[Staff_collecting_engraver], page 448.
Music types accepted: tempo-change-event (page 58),
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature,
    etc.).
  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): MetronomeMark (page 585).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Paper_column_engraver (page 440)
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: break-event (page 49), and label-event (page 52),

Properties (read)

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly re-
quested by the user.

Properties (write)

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

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, except if explicitly re-
quested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): NonMusicalPaperColumn
(page 593), and PaperColumn (page 600).

Parenthesis_engraver (page 440)
Parenthesize objects whose parenthesize property is #t.
This engraver creates the following layout object(s): Parentheses (page 601).

Repeat_acknowledge_engraver (page 443)
This translator adds entries to repeatCommands for events generated by \\repeat
volta.
Music types accepted: volta-repeat-end-event (page 59), and volta-repeat-
start-event (page 59),
Properties (write)

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args...)', but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as 'start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

Show_control_points_engraver (page 446)
Create grobs to visualize control points of Bézier curves (ties and slurs) for ease of tweaking.
This engraver creates the following layout object(s): ControlPoint (page 519), and ControlPolygon (page 520).

Spacing_engraver (page 447)
Make a SpacingSpanner and do bookkeeping of shortest starting and playing notes.
Music types accepted: spacing-section-event (page 56),

Properties (read)

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

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): SpacingSpanner (page 626).

Spanner_tracking_engraver (page 448)
Helper for creating spanners attached to other spanners. If a spanner has the sticky-grob-interface, the engraver tracks the spanner contained in its sticky-host object. When the host ends, the sticky spanner attached to it has its end announced too.

Staff_collecting_engraver (page 448)
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.
Stanza_number_align_engraver (page 449)
This engraver ensures that stanza numbers are neatly aligned.

System_start_delimiter_engraver (page 450)
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 items (clef, key signature, etc.).

- 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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

Timing_translator (page 454)
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.
Music types accepted: alternative-event (page 48), bar-event (page 49), and fine-event (page 51),
Properties (read)

- alternativeNumberingStyle (symbol)
  The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

- 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 time-keeping, 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)

`alternativeNumber` (non-negative, exact integer)
When set, the index of the current alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple volte.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`currentBarNumber` (integer)
Contains the current bar number. This property is incremented at every bar line.

`internalBarNumber` (integer)
Contains the current bar number. This property is used for internal time-keeping, 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.

`measureStartNow` (boolean)
True at the beginning of a measure.

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, '(4 . 4) is a 4/4 time signature.

Tweak_engraver (page 456)
Read the tweaks property from the originating event, and set properties.

Vertical_align_engraver (page 456)
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): StaffGrouper (page 630), and VerticalAlignment (page 668).

Volta_engraver (page 456)
Make volta brackets.

Music types accepted: dal-segno-event (page 50), fine-event (page 51), and volta-span-event (page 59),

Properties (read)

`currentCommandColumn` (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).
repeatCommands (list)
  A list of commands related to volta-style repeats. In general, each ele-
  ment is a list, '(command args...)', but a command with no arguments
  may be abbreviated to a symbol; e.g., '((start-repeat)) may be given
  as '(start-repeat).

end-repeat
  End a repeated section.

start-repeat
  Start a repeated section.

volta text
  If text is markup, start a volta bracket with that label; if text is #f,
  end a volta bracket.

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): VoltaBracket (page 671), and
VoltaBracketSpanner (page 672).

2.1.34 StandaloneRhythmStaff

A Staff-level context for use by \markup \rhythm.

This context also accepts commands for the following context(s): Staff (page 287), and
Staff (page 287).

This context creates the following layout object(s): BarLine (page 486), BreathingSign
(page 502), CaesuraScript (page 504), DotColumn (page 531), InstrumentName (page 559),
LedgerLineSpanner (page 570), StaffHighlight (page 631), StaffSpacing (page 632),
StaffSymbol (page 632), and VerticalAxisGroup (page 668).

This context sets the following properties:
• Set context property createSpacing to #t.
• Set context property instrumentName to '() .
• Set context property localAlterations to '() .
• Set context property shortInstrumentName to '() .
• Set context property squashedPosition to 0 .
• Set context property squashedPosition to 1 .
• Set grob property line-count in StaffSymbol (page 632), to 0 .
• Set grob property line-count in StaffSymbol (page 632), to 1 .
• Set grob property neutral-direction in Beam (page 495), to 1 .
• Set grob property neutral-direction in Stem (page 634), to 1 .
• Set grob property staff-padding in VoltaBracket (page 671), to 3 .

This is not a 'Bottom' context; search for such a one will commence after creating an implicit
context of type StandaloneRhythmVoice (page 331).

Context StandaloneRhythmStaff can contain CueVoice (page 97), NullVoice (page 228),
StandaloneRhythmVoice (page 331), and Voice (page 390).
This context is built from the following engraver(s):

Axis\_group\_engraver (page 403)
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 items (clef, key signature, etc.).

- 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): VerticalAxisGroup (page 668).

Bar\_engraver (page 404)
Create bar lines for various commands, including $\bar{}$.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

- caesuraType (list)
  An alist

  $\langle (\text{bar-line} \ . \ \text{bar-type})$
  $\langle \text{breath} \ . \ \text{breath-type})$
  $\langle \text{scripts} \ . \ \text{script-type...})$
  $\langle \text{underlying-bar-line} \ . \ \text{bar-type})\rangle$

  specifying which breath mark, bar line, and scripts to create at $\caesura$. All entries are optional.

  bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

- caesuraTypeTransform (procedure)
  An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

  The first argument is the context.

  The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

**doubleRepeatBarType (string)**

Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:...:’.

**doubleRepeatSegnoBarType (string)**

Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.S.|:’.

**endRepeatBarType (string)**

Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

**endRepeatSegnoBarType (string)**

Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.S’.

**fineBarType (string)**

Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

**fineSegnoBarType (string)**

Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.S’.

**fineStartRepeatSegnoBarType (string)**

Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.|:’.

**forbidBreakBetweenBarLines (boolean)**

If set to true, Bar_engraver forbids line breaks where there is no bar line.

**measureBarType (string)**

Bar line to insert at a measure boundary.

**repeatCommands (list)**

A list of commands related to volta-style repeats. In general, each element is a list, '((command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

**end-repeat**

End a repeated section.

**start-repeat**

Start a repeated section.

**volta text**

If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

**sectionBarType (string)**

Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.
segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
The current bar line type, or ‘()’ if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

currentUser (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
Notate a short break in sound that does not shorten the previous note.

Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.

If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),

Properties (read)

breathMarkDefinitions (list)
The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

caesuraType (list)
An alist

  ((bar-line . bar-type)
   (breath . breath-type)
   (scripts . script-type...))
specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Dot_column_engraver (page 418)
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): DotColumn (page 531).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

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.

tvocalName (markup)
    Name of a vocal line.

This engraver creates the following layout object(s): InstrumentName (page 559).

Ledger_line_engraver (page 431)
    Create the spanner to draw ledger lines, and notices objects that need ledger lines.
    This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Output_property_engraver (page 439)
    Apply a procedure to any grob acknowledged.
    Music types accepted: apply-output-event (page 48),

Pitch_squash_engraver (page 443)
    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.

Separating_line_group_engraver (page 445)
    Generate objects for computing spacing parameters.
    Properties (read)

    createSpacing (boolean)
        Create StaffSpacing objects? Should be set for staves.
    Properties (write)

    hasStaffSpacing (boolean)
        True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 632).

Staff_highlight_engraver (page 449)
    Highlights music passages.
    Music types accepted: staff-highlight-event (page 56),
    Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
    Create the constellation of five (default) staff lines.
    Music types accepted: staff-span-event (page 57),
    This engraver creates the following layout object(s): StaffSymbol (page 632).
2.1.35 StandaloneRhythmVoice

A Voice-level context for use by \markup \rhythm. This context also accepts commands for the following context(s): Voice (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Fingering (page 545), Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), LigatureBracket (page 573), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), Stem (page 634), StemStub (page 636), StemTremolo (page 637), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), and VoiceFollower (page 670).

This context sets the following properties:

- Set grob property direction in Stem (page 634), 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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.

Music types accepted: arpeggio-event (page 49),

This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 49),

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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags
instead of beams.
Music types accepted: beam-event (page 49),
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): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.
Music types accepted: bend-after-event (page 49),
Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current
timestep.

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

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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)

breathMarkType (symbol)
The type of BreathingSign to create at \breathe.
This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514), and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 \[Dots\], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
    quested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat
(page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner
(page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-
event (page 49), and span-dynamic-event (page 56),
Properties (read)

\texttt{crescendoSpanner} (symbol)

The type of spanner to be used for crescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin crescendo is used.

\texttt{crescendoText} (markup)

The text to print at start of non-hairpin crescendo, i.e., ‘cresc.’.

\texttt{currentMusicalColumn} (graphical (layout) object)

Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

\texttt{decrescendoSpanner} (symbol)

The type of spanner to be used for decrescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin decrescendo is used.

\texttt{decrescendoText} (markup)

The text to print at start of non-hairpin decrescendo, i.e., ‘dim.’.

This engraver creates the following layout object(s): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

\texttt{Finger_glide_engraver} (page 422)

Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),

This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

\texttt{Fingering_engraver} (page 422)

Create fingering scripts.

Music types accepted: fingering-event (page 51),

This engraver creates the following layout object(s): Fingering (page 545).

\texttt{Font_size_engraver} (page 422)

Put fontSize into font-size grob property.

Properties (read)

\texttt{fontSize} (number)

The relative size of all grobs in a context.

\texttt{Forbid_line_break_engraver} (page 423)

Forbid line breaks when note heads are still playing at some point.

Properties (read)

\texttt{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)

\texttt{forbidBreak} (boolean)

If set to #t, prevent a line break at this point, except if explicitly requested by the user.

\texttt{Glissando_engraver} (page 424)

Engrave glissandi.

Music types accepted: glissando-event (page 51),
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 nu-
      mber of note-heads in the two note columns between which the glissandi
      occur.

This engraver creates the following layout object(s): Glissando (page 552).

Grace_auto_beam_engraver (page 424)
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: beam-forbid-event (page 49),
Properties (read)
    autoBeaming (boolean)
    If set to true then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).

Grace_beam_engraver (page 425)
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: beam-event (page 49),
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): Beam (page 495).

Grace_engraver (page 425)
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.

Grob_pq_engraver (page 426)
Administrate when certain grobs (e.g., note heads) stop playing.
Properties (read)
    busyGrobs (list)
    A queue of \(\text{end-moment} . \text{grob}\) 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)

busyGrosbs (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.).

Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

instrumentCueName (markup)
The name to print if another instrument is to be taken.
This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),
This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

Ligature_bracket_engraver (page 431)
Handle Ligature_events by engraving Ligature brackets.
Music types accepted: ligature-event (page 52),
This engraver creates the following layout object(s): LigatureBracket (page 573).

Multi_measure_rest_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.
Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)

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

internalBarNumber (integer)
Contains the current barnumber. This property is used for internal time-keeping, among others by the Accidental_engraver.

measureStartNow (boolean)
True at the beginning of a measure.

restNumberThreshold (number)
If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and MultiMeasureRestText (page 592).

New_fingering_engraver (page 437)
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): Fingering (page 545), Script (page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)
   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): VoiceFollower (page 670).

Note_heads_engraver (page 438)
   Generate note heads.

Music types accepted: note-event (page 54),

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): NoteHead (page 596).

Note_spacing_engraver (page 439)
   Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
   Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
   Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.

Music types accepted: note-event (page 54), and part-combine-event (page 54),

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?

soloIIIText (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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
  Make whole measure repeats.
  Music types accepted: percent-event (page 54),
  Properties (read)

  countPercentRepeats (boolean)
    If set, produce counters for percent repeats.

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

  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
  Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
  Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
  This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
  Print the bracketed note head after a note head with trill.
  This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
  Create repeat ties.
  Music types accepted: repeat-tie-event (page 55),
  This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
  Engrave rests.
  Music types accepted: rest-event (page 55),
  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): Rest (page 611).

**Rhythmic_column_engraver (page 445)**
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

**Script_column_engraver (page 445)**
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

**Script_engraver (page 445)**
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

**Slash_repeat_engraver (page 446)**
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

**Slur_engraver (page 446)**
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

**Spanner_break_forbid_engraver (page 448)**
Forbid breaks in certain spanners.

**Stem_engraver (page 450)**
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)

  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
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.

This engraver creates the following layout object(s): Flag (page 547), Stem
(page 634), StemStub (page 636), and StemTremolo (page 637).

Text_engraver (page 452)
    Create text scripts.
    Music types accepted: text-script-event (page 58),
    This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_engraver (page 452)
    Create text spanner from an event.
    Music types accepted: text-span-event (page 58),
    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): TextSpanner (page 652).

Tie_engraver (page 452)
    Generate ties between note heads of equal pitch.
    Music types accepted: tie-event (page 58),
    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): Tie (page 653), and
    TieColumn (page 655).

Trill_spanner_engraver (page 455)
    Create trill spanners.
    Music types accepted: trill-span-event (page 58),
    Properties (read)
        currentCommandColumn (graphical (layout) object)
            Grob that is X-parent to all current breakable items (clef, key signature,
            etc.).

        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): TrillSpanner (page 661).

Tuplet_engraver (page 455)
Catch tuplet events and generate appropriate bracket.
Music types accepted: tuplet-span-event (page 58),
Properties (read)

\[\text{tupletFullLength} \text{ (boolean)}\]
If set, the tuplet is printed up to the start of the next note.

\[\text{tupletFullLengthNote} \text{ (boolean)}\]
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): TupletBracket (page 662),
and TupletNumber (page 664).

2.1.36 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 (page 287).

This context creates the following layout object(s): BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), BreathingSign (page 502), CaesuraScript (page 504), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), LedgerLineSpanner (page 570), NoteCollision (page 595), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642), TimeSignature (page 655), UnaCordaPedal (page 665), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:
- Set context property autoBeaming to #f.
- Set context property clefGlyph to "clefs.tab".
- Set context property clefPosition to 0.
- Set context property createSpacing to #t.
- Set context property handleNegativeFrets to 'recalculate.
- Set context property ignoreFiguredBassRest to #f.
- Set context property instrumentName to '().
- Set context property localAlterations to '().
- Set context property ottavationMarkups to:
  '(((4 . "29")
  (3 . "22")
  (2 . "15")
  (1 . "8")
  (-1 . "8")
  (-2 . "15")
  (-3 . "22")
  (-4 . "29"))
• Set context property restrainOpenStrings to #f.
• Set context property shortInstrumentName to '().
• Set grob property after-line-breaking in RepeatTie (page 610), to repeat-tie::handle-tab-note-head.
• Set grob property after-line-breaking in Tie (page 653), to tie::handle-tab-note-head.
• Set grob property avoid-note-head in Stem (page 634), to #t.
• Set grob property beam-thickness in Beam (page 495), to 0.32.
• Set grob property beam-thickness in StemTremolo (page 637), to 0.32.
• Set grob property beam-width in StemTremolo (page 637), to stem-tremolo::calc-tab-width.
• Set grob property bound-details.left in Glissando (page 552), to: '((attach-dir . 1) (padding . 0.3))
• Set grob property bound-details.right in Glissando (page 552), to: '((attach-dir . -1) (padding . 0.3))
• Set grob property control-points in Slur (page 621), to #cumpure-pure-container #<procedure 5575a263a040 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:2545:16 (grob)> #<procedure 5575a263a020 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:2547:16 (grob . rest)> >.
• Set grob property details in Stem (page 634), to: '(((lengths 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 Glissando (page 552), to glissando::calc-tab-extra-dy.
• Set grob property glyph-name in TabNoteHead (page 648), to tab-note-head::calc-glyph-name.
• Set grob property ignore-collision in NoteColumn (page 595), to #t.
• Set grob property length-fraction in Beam (page 495), to 0.62.
• Set grob property length-fraction in StemTremolo (page 637), to #<procedure 5575a37c7780 at ice-9/eval.scm:333:13 (a)>.
• Set grob property no-stem-extend in Stem (page 634), to #t.
• Set grob property staff-space in StaffSymbol (page 632), to 1.5.
• Set grob property stencil in Arpeggio (page 483), to #f.
• Set grob property stencil in Beam (page 495), to #f.
• Set grob property stencil in Clef (page 510), to clef::print-modern-tab-if-set.
• Set grob property stencil in Dots (page 531), to #f.
• Set grob property stencil in DynamicTextSpanner (page 541), to #f.
• Set grob property stencil in DynamicText (page 539), to #f.
• Set grob property stencil in Flag (page 547), to #f.
• Set grob property stencil in Glissando (page 552), to glissando::draw-tab-glissando.
• Set grob property stencil in Hairpin (page 555), to #f.
• Set grob property stencil in LaissezVibrerTie (page 569), to #f.
• Set grob property stencil in MultiMeasureRestNumber (page 589), to #f.
• Set grob property stencil in MultiMeasureRestScript (page 590), to #f.
• Set grob property stencil in MultiMeasureRestText (page 592), to #f.
• Set grob property stencil in MultiMeasureRest (page 587), to #f.
• Set grob property stencil in PhrasingSlur (page 604), to #f.
• Set grob property stencil in RepeatTie (page 610), to #f.
• Set grob property stencil in Rest (page 611), to #f.
• Set grob property stencil in Script (page 613), to #f.
• Set grob property stencil in StemTremolo (page 637), to #f.
• Set grob property stencil in Stem (page 634), to #f.
• Set grob property stencil in TabNoteHead (page 648), to tab-note-head::whiteout-if-style-set.
• Set grob property stencil in TextScript (page 650), to #f.
• Set grob property stencil in TextSpanner (page 652), to #f.
• Set grob property stencil in Tie (page 653), to #f.
• Set grob property stencil in TimeSignature (page 655), to #f.
• Set grob property stencil in TupletBracket (page 662), to #f.
• Set grob property stencil in TupletNumber (page 664), to #f.
• Set grob property style in Flag (page 547), to 'no-flag.

This is not a 'Bottom' context; search for such a one will commence after creating an implicit context of type TabVoice (page 352).

Context TabStaff can contain CueVoice (page 97), NullVoice (page 228), and TabVoice (page 352).

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

**Alteration_glyph_engraver** (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context's alterationGlyphs property, when defined.

Properties (read)

  alterationGlyphs (list)

  Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

**Axis_group_engraver** (page 403)
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 items (clef, key signature, etc.).

  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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)

Create bar lines for various commands, including \\bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).

Properties (read)

caesuraType (list)

An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)

An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

The first argument is the context.

The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)

Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:\\:\\’.

doubleRepeatSegnoBarType (string)

Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:\\.S\.\:\’.

endRepeatBarType (string)

Bar line to insert at the end of a \repeat volta. The default is ‘:\:\’.

endRepeatSegnoBarType (string)

Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:\\.S’.
fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the
repeat bar line takes precedence and this value is appended to it as an
annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The de-
default is ‘|.S’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the
start of a \repeat volta. The default is ‘|.S.|:’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar
line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each ele-
ment is a list, ‘(command args...), but a command with no arguments
may be abbreviated to a symbol; e.g., ‘((start-repeat)) may be given
as ‘(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f,
end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the
repeat bar line takes precedence and this value is appended to it as an
annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.|:’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a
\repeat volta. The default is ‘S.|:’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line
would normally appear, for example at the end of a system broken in
mid measure where the next system begins with a segno. Where there is
also a repeat bar line, the repeat bar line takes precedence and this value
is appended to it as an annotation. The default is ‘||’.
whichBar (string)
   The current bar line type, or '()' if there is no bar line. Setting this explicitely in user code is deprecated. Use \bar or related commands to set it.

Properties (write)

currentBarLine (graphical (layout) object)
   Set to the BarLine that Bar_engraver has created in the current timestep.

forbidBreak (boolean)
   If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Caesura_engraver (page 410)
   Notate a short break in sound that does not shorten the previous note.

Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.

If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),

Properties (read)

breathMarkDefinitions (list)
   The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

caesuraType (list)
   An alist

   ((bar-line . bar-type)
    (breath . breath-type)
    (scripts . script-type...)
    (underlying-bar-line . bar-type))

   specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

   bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.

   The first argument is the context.

   The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Clef_engraver (page 413)
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.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and ClefModifier (page 513).

Collision_engraver (page 413)
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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
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.
Chapter 2: Translation

`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.

`forbidBreak` (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

`forceBreak` (boolean)
Set to #t when an event forcing a line break was heard.

`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): ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

`Dot_column_engraver` (page 418)
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): DotColumn (page 531).

`Figured_bass_engraver` (page 421)
Make figured bass numbers.

Music types accepted: bass-figure-event (page 49), and rest-event (page 55).

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): BassFigure (page 491), BassFigureAlignment (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.
This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that will fix the collisions.
This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
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.).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

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): \texttt{InstrumentName} (page 559).

\texttt{Ledger\_line\_engraver} (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines.
This engraver creates the following layout object(s): \texttt{LedgerLineSpanner} (page 570).

\texttt{Merge\_mmrest\_numbers\_engraver} (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

\texttt{Non\_musical\_script\_column\_engraver} (page 437)
Find potentially colliding non-musical scripts and put them into a \texttt{ScriptColumn} object; that will fix the collisions.
This engraver creates the following layout object(s): \texttt{ScriptColumn} (page 614).

\texttt{Output\_property\_engraver} (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: \texttt{apply\_output\_event} (page 48),

\texttt{Piano\_pedal\_align\_engraver} (page 442)
Align piano pedal symbols and brackets.
Properties (read)
\begin{itemize}
  \item \texttt{currentCommandColumn} (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
\end{itemize}
This engraver creates the following layout object(s): \texttt{SostenutoPedalLineSpanner} (page 624), \texttt{SustainPedalLineSpanner} (page 642), and \texttt{UnaCordaPedalLineSpanner} (page 666).

\texttt{Piano\_pedal\_engraver} (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: \texttt{sostenuto\_event} (page 56), \texttt{sustain\_event} (page 57), and \texttt{una\_corda\_event} (page 58),
Properties (read)
\begin{itemize}
  \item \texttt{currentCommandColumn} (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  \item \texttt{pedalSostenutoStrings} (list)
    See \texttt{pedalSustainStrings}.
  \item \texttt{pedalSostenutoStyle} (symbol)
    See \texttt{pedalSustainStyle}.
  \item \texttt{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.
  \item \texttt{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): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)
    Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
    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): RestCollision (page 612).

Script_row_engraver (page 445)
    Determine order in horizontal side position elements.
    This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
    Generate objects for computing spacing parameters.
    Properties (read)
    createSpacing (boolean)
        Create StaffSpacing objects? Should be set for staves.
    Properties (write)
    hasStaffSpacing (boolean)
        True if currentCommandColumn contains items that will affect spacing.

    This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
    Create a StaffEllipsis when skipTypesetting is used.
    Properties (read)
    skipTypesetting (boolean)
        If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

    This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
    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_highlight_engraver (page 449)
Highlights music passages.
Music types accepted: staff-highlight-event (page 56),
Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_symbol_engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

Tab_staff_symbol_engraver (page 451)
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): StaffSymbol (page 632).

Time_signature_engraver (page 453)
Create a Section 3.1.146 [TimeSignature], page 655, whenever
timeSignatureFraction changes.
Music types accepted: time-signature-event (page 58),
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): TimeSignature (page 655).

2.1.37 TabVoice
Context for drawing notes in a Tab staff.

This context also accepts commands for the following context(s): Voice (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495),
BendAfter (page 497), BendSpanner (page 498), BreathingSign (page 502), ClusterSpanner
(page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots
(page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533),
DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText
(page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Flag
(page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560),
LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), LigatureBracket
(page 573), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589),
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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.
Music types accepted: arpeggio-event (page 49),
This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.
Music types accepted: beam-forbid-event (page 49),
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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams.
Music types accepted: beam-event (page 49),
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): Beam (page 495).

Bend_ engraver (page 409)  
Create fall spanners.

Music types accepted: bend-after-event (page 49),

Properties (read)  

currentBarLine (graphical (layout) object)  
Set to the BarLine that Bar_ engraver has created in the current timestep.

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

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): BendAfter (page 497).

Bend_spanner_ engraver (page 410)  
Engraver to print a BendSpanner.

Music types accepted: bend-span-event (page 49), note-event (page 54), and string-number-event (page 57),

Properties (read)  

stringFretFingerList (list)  
A list containing three entries. In TabVoice and FretBoards they determine the string, fret and finger to use

supportNonIntegerFret (boolean)  
If set in Score the TabStaff will print micro-tones as ‘2\frac{1}{2}’

Properties (write)  

stringFretFingerList (list)  
A list containing three entries. In TabVoice and FretBoards they determine the string, fret and finger to use

supportNonIntegerFret (boolean)  
If set in Score the TabStaff will print micro-tones as ‘2\frac{1}{2}’

This engraver creates the following layout object(s): BendSpanner (page 498).

Breathing_sign_ engraver (page 410)  
Notate breath marks.

Music types accepted: breathing-event (page 50),

Properties (read)  

breathMarkType (symbol)  
The type of BreathingSign to create at \breathe.
This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514), and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
    quested by the user.
This engraver creates the following layout object(s): DoublePercentRepeat
(page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner
(page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-
  event (page 49), and span-dynamic-event (page 56),
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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),

This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.

Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Forbid_line_break_engraver (page 423)
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, except if explicitly requested by the user.

Glissando_engraver (page 424)
Engrave glissandi.

Music types accepted: glissando-event (page 51),

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): Glissando (page 552).

**Grace_auto_beam_engraver** (page 424)
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: beam-forbid-event (page 49),

**Properties (read)**

autoBeaming (boolean)
If set to true then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).

**Grace_beam_engraver** (page 425)
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: beam-event (page 49),

**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): Beam (page 495).

**Grace_engraver** (page 425)
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.

**Grob_pq_engraver** (page 426)
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.).
Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)
  instrumentCueName (markup)
  The name to print if another instrument is to be taken.
  This property is deprecated
This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),
This engraver creates the following layout object(s): LaissezVibrerTie (page 569),
and LaissezVibrerTieColumn (page 570).

Ligature_bracket_engraver (page 431)
Handle Ligature_events by engraving Ligature brackets.
Music types accepted: ligature-event (page 52),
This engraver creates the following layout object(s): LigatureBracket (page 573).

Multi_measure_rest_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow
and internalBarNumber to determine what number to print over the Section 3.1.88
[MultiMeasureRest], page 587.
Music types accepted: multi-measure-articulation-event (page 53),
multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)
  currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  internalBarNumber (integer)
  Contains the current barnumber. This property is used for internal time-keeping, among others by the Accidental_engraver.
  measureStartNow (boolean)
  True at the beginning of a measure.
  restNumberThreshold (number)
  If a multimeasure rest has more measures than this, a number is printed.
This engraver creates the following layout object(s): MultiMeasureRest (page 587),
MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and
MultiMeasureRestText (page 592).

Note_head_line_engraver (page 438)
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): VoiceFollower (page 670).
Note_spacing_engraver (page 439)
Generate NoteSpacing, an object linking horizontal lines for use in spacing.
This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.
Music types accepted: note-event (page 54), and part-combine-event (page 54),
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?
  soloIIIText (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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)
  countPercentRepeats (boolean)
  If set, produce counters for percent repeats.
  currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
This engraver creates the following layout object(s): PhrasingSlur (page 604).
Repeat_tie_engraver (page 444)

Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)

Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).

Rhythmic_column_engraver (page 445)

Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)

Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)

Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)

Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

Slur_engraver (page 446)

Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).
Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Stem_engraver (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)

    currentBarLine (graphical (layout) object)
    Set to the BarLine that Bar_engraver has created in the current timestep.

    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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

Tab_note_heads_engraver (page 451)
Generate one or more tablature note heads from event of type NoteEvent.

Music types accepted: fingering-event (page 51), note-event (page 54), and string-number-event (page 57),
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.

    maximumFretStretch (number)
    Don’t allocate frets further than this from specified frets.

    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 argu-
ments: 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): TabNoteHead (page 648).

Tab_tie_follow_engraver (page 452)
Adjust TabNoteHead properties when a tie is followed by a slur or glissando.

Text_engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Text_spanner_engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
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): TextSpanner (page 652).

Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
Properties (read)

    skipTypesetting (boolean)
    If true, no typesetting is done, speeding up the interpretation phase.
    Useful for debuging 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): Tie (page 653), and
TieColumn (page 655).
Trill_spanner_engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)

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

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): TrillSpanner (page 661).

Tuplet_engraver (page 455)
Catch tuplet events and generate appropriate bracket.
Music types accepted: tuplet-span-event (page 58),
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): TupletBracket (page 662), and TupletNumber (page 664).

2.1.38 VaticanaLyrics
Same as Lyrics context, except that it provides a hyphenation style (a single, flush-left hyphen between two syllables) as used in the notational style of Editio Vaticana.

This context also accepts commands for the following context(s): Lyrics (page 199).

This context creates the following layout object(s): InstrumentName (page 559),
LyricExtender (page 574), LyricHyphen (page 575), LyricSpace (page 578), LyricText (page 579), StanzaNumber (page 633), VerticalAxisGroup (page 668), and VowelTransition (page 673).

This context sets the following properties:
• Set context property instrumentName to '().
• Set context property lyricRepeatCountFormatter to #<procedure 5575a3702120 at /build/out/share/lilypond/current/scm/lily/translation-functions.scm:210:4 (context repeat-count)>.
• Set context property searchForVoice to #f.
• Set context property shortInstrumentName to '().
• Set grob property bar-extent in BarLine (page 486), to:
  '(-0.05 . 0.05)
• Set grob property font-series in LyricHyphen (page 575), to 'medium.
• Set grob property font-size in InstrumentName (page 559), to 1.0.
• Set grob property font-size in LyricHyphen (page 575), to -4.
• Set grob property font-size in LyricText (page 579), to -4.
• Set grob property nonstaff-nonstaff-spacing in VerticalAxisGroup (page 668), to:

'((basic-distance . 0)
 (minimum-distance . 2.8)
 (padding . 0.2)
 (stretchability . 0))

• Set grob property nonstaff-relatedstaff-spacing in VerticalAxisGroup (page 668), to:

'((basic-distance . 5.5)
 (padding . 0.5)
 (stretchability . 1))

• Set grob property nonstaff-unrelatedstaff-spacing.padding in VerticalAxisGroup (page 668), to 1.5.

• Set grob property remove-empty in VerticalAxisGroup (page 668), to #t.

• Set grob property remove-first in VerticalAxisGroup (page 668), to #t.

• Set grob property self-alignment-Y in InstrumentName (page 559), to #f.

• Set grob property staff-affinity in VerticalAxisGroup (page 668), to 1.

• Set grob property stencil in LyricHyphen (page 575), to lyric-hyphen::vaticana-style.

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):

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).

  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): VerticalAxisGroup (page 668).

Extender_engraver (page 421)
Create lyric extenders.
Music types accepted: completize-extender-event (page 50), and extender-event (page 51),
Properties (read)

  extendersOverRests (boolean)
  Whether to continue extenders as they cross a rest.
This engraver creates the following layout object(s): LyricExtender (page 574).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)
  fontSize (number)
The relative size of all grobs in a context.

Hyphen_engraver (page 427)
Create lyric hyphens, vowel transitions and distance constraints between words.
Music types accepted: hyphen-event (page 52), and vowel-transition-event (page 59),
This engraver creates the following layout object(s): LyricHyphen (page 575),
LyricSpace (page 578), and VowelTransition (page 673).

Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

  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): InstrumentName (page 559).

Lyric_engraver (page 431)
Engrave text for lyrics.
Music types accepted: lyric-event (page 52),
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): LyricText (page 579).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.
Stanza_number_engraver (page 450)
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): StanzaNumber (page 633).

2.1.39 VaticanaStaff

Configure division commands such as \section to create Divisio grobs rather than BarLine grobs. This does not affect measure bar lines or the properties of the grobs themselves.

This context also accepts commands for the following context(s): Staff (page 287).

This context creates the following layout object(s): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), AccidentalSuggestion (page 478), BarLine (page 486), BassFigure (page 491), BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494), Clef (page 510), ClefModifier (page 513), CueClef (page 521), CueEndClef (page 524), Custos (page 527), Divisio (page 528), DotColumn (page 531), FingeringColumn (page 547), InstrumentName (page 559), KeyCancellation (page 563), KeySignature (page 565), LedgerLineSpanner (page 570), NoteCollision (page 595), OttavaBracket (page 599), PianoPedalBracket (page 606), RestCollision (page 612), ScriptColumn (page 614), ScriptRow (page 614), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624), StaffEllipsis (page 628), StaffHighlight (page 631), StaffSpacing (page 632), StaffSymbol (page 632), SustainPedal (page 641), SustainPedalLineSpanner (page 642), UnaCordaPedal (page 665), UnaCordaPedalLineSpanner (page 666), and VerticalAxisGroup (page 668).

This context sets the following properties:

• Set context property alterationGlyphs to: '((-1/2 . "accidentals.vaticanaM1")
  (0 . "accidentals.vaticana0")
  (1/2 . "accidentals.mensural11"))

• Set context property autoAccidentals to: '(Staff #(procedure 5575a36e3080 at /build/out/share/lilypond/current/scm/lily/music-functions.scm:1681:0
  • Set context property autoCautionaries to: '().

• Set context property caesuraTypeTransform to caesura-to-bar-line-or-divisio.

• Set context property caesuraTypeTransform to caesura-to-divisio.

• Set context property caesuraType to: '((breath . varcomma))

• Set context property clefGlyph to "clefs.vaticana.do".

• Set context property clefPosition to 1.

• Set context property clefTransposition to 0.

• Set context property createSpacing to #t.

• Set context property doubleRepeatBarType to "||".

• Set context property doubleRepeatBarType to '().

• Set context property doubleRepeatSegnoBarType to "S-||".
• Set context property doubleRepeatSegnoBarType to "S-||".
• Set context property endRepeatBarType to "||".
• Set context property endRepeatBarType to '()'.
• Set context property endRepeatSegnoBarType to "S-||".
• Set context property endRepeatSegnoBarType to "S-||".
• Set context property extraNatural to #f.
• Set context property fineBarType to "".
• Set context property fineBarType to "||".
• Set context property fineSegnoBarType to "S-||".
• Set context property fineSegnoBarType to "S-||".
• Set context property fineStartRepeatSegnoBarType to "S-||".
• Set context property fineStartRepeatSegnoBarType to "S-||".
• Set context property forbidBreakBetweenBarLines to #f.
• Set context property ignoreFiguredBassRest to #f.
• Set context property instrumentName to '()'.
• Set context property localAlterations to '()'.
• Set context property measureBarType to '()'.
• Set context property middleCClefPosition to 1.
• Set context property middleCPosition to 1.
• Set context property ottavationMarkups to:

  '
((4 "29")
 (3 "22")
 (2 "15")
 (1 "8")
 (-1 "8")
 (-2 "15")
 (-3 "22")
 (-4 "29"))

• Set context property printKeyCancellation to #f.
• Set context property sectionBarType to "".
• Set context property sectionBarType to "||".
• Set context property segnoBarType to "S-||".
• Set context property segnoBarType to "S-||".
• Set context property shortInstrumentName to '()'.
• Set context property startRepeatBarType to "||".
• Set context property startRepeatBarType to '()'.
• Set context property startRepeatSegnoBarType to "S-||".
• Set context property startRepeatSegnoBarType to "S-||".
• Set context property underlyingRepeatBarType to "".
• Set context property underlyingRepeatBarType to "||".
• Set grob property extra-spacing-height in BreathingSign (page 502), to
  item::extra-spacing-height-including-staff.
• Set grob property extra-spacing-width in BreathingSign (page 502), to:
  '(-1.0 . 0.0)
• Set grob property \texttt{font-size} in \texttt{BreathingSign} (page 502), to \texttt{-2}.
• Set grob property \texttt{font-size} in \texttt{Divisio} (page 528), to \texttt{-2}.
• Set grob property \texttt{hair-thickness} in \texttt{BarLine} (page 486), to \texttt{0.6}.
• Set grob property \texttt{ledger-line-thickness} in \texttt{StaffSymbol} (page 632), to :
  \texttt{'(1 . 0)}
• Set grob property \texttt{length-fraction} in \texttt{LedgerLineSpanner} (page 570), to \texttt{0.9}.
• Set grob property \texttt{line-count} in \texttt{StaffSymbol} (page 632), to \texttt{4}.
• Set grob property \texttt{neutral-direction} in \texttt{Custos} (page 527), to \texttt{-1}.
• Set grob property \texttt{neutral-position} in \texttt{Custos} (page 527), to \texttt{3}.
• Set grob property \texttt{space-alist.clef} in \texttt{LeftEdge} (page 571), to :
  \texttt{'(extra-space . 0)}
• Set grob property \texttt{space-alist.custos} in \texttt{BarLine} (page 486), to :
  \texttt{'(minimum-space . 0.7)}
• Set grob property \texttt{space-alist.first-note} in \texttt{Clef} (page 510), to :
  \texttt{'(minimum-fixed-space . 1.4)}
• Set grob property \texttt{space-alist.right-edge} in \texttt{Custos} (page 527), to :
  \texttt{'(extra-space . 0)}
• Set grob property \texttt{style} in \texttt{Custos} (page 527), to \texttt{'vaticana}.
• Set grob property \texttt{style} in \texttt{Dots} (page 531), to \texttt{'vaticana}.
• Set grob property \texttt{thick-thickness} in \texttt{BarLine} (page 486), to \texttt{1.8}.
• Set grob property \texttt{thickness} in \texttt{BreathingSign} (page 502), to \texttt{1}.
• Set grob property \texttt{thickness} in \texttt{Divisio} (page 528), to \texttt{1}.
• Set grob property \texttt{thickness} in \texttt{StaffSymbol} (page 632), to \texttt{0.6}.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type \texttt{VaticanaVoice} (page 380).

Context \texttt{VaticanaStaff} can contain \texttt{CueVoice} (page 97), \texttt{NullVoice} (page 228), and \texttt{VaticanaVoice} (page 380).

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

\begin{itemize}
  \item \texttt{Accidental_engraver} (page 400)
    \begin{itemize}
      \item 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 \texttt{	extbackslash override} them at Voice.
      \item Properties (read)
      \begin{itemize}
        \item \texttt{accidentalGrouping} (symbol)
          \begin{itemize}
            \item If set to \texttt{'voice}, accidentals on the same note in different octaves may be horizontally staggered if in different voices.
          \end{itemize}
        \item \texttt{autoAccidentals} (list)
          \begin{itemize}
            \item List of different ways to typeset an accidental.
              \begin{itemize}
                \item For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.
                \item Each entry in the list is either a symbol or a procedure.
              \end{itemize}
            \item \texttt{symbol}
              \begin{itemize}
                \item The symbol is the name of the context in which the following rules are to be applied. For example, if \texttt{context} is Section \texttt{“Score”} in \texttt{Internals Reference} then all staves share accidentals, and if \texttt{context}
              \end{itemize}
          \end{itemize}
      \end{itemize}
    \end{itemize}
\end{itemize}
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.

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 bar number. This property is used for internal time-keeping, 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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477), and AccidentalSuggestion (page 478).
Chapter 2: Translation

Alteration_glyph_engraver (page 401)
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value of the context’s alterationGlyphs property, when defined.
Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Axis_group_engraver (page 403)
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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Bar_engraver (page 404)
Create bar lines for various commands, including \bar.
If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.
Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50), coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51), section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59).
Properties (read)

caesuraType (list)
An alist

((bar-line . bar-type)
(breath . breath-type)
(scripts . script-type...)
(underlying-bar-line . bar-type))
specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.

The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.

The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘:...’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘:|.|’.

endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:|.’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:|.|’.

fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘|.’.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is ‘|.|’.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is ‘|.|’.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_ engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, ‘(command args...), but a command with no arguments may be abbreviated to a symbol; e.g., ‘((start-repeat)) may be given as ‘(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.
volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is ‘S’.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is ‘.||’.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.||’.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

whichBar (string)
The current bar line type, or ’() if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

Properties (write)
currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Clef_engraver (page 413)
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.

forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly re-
  quested by the user.

forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

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): Clef (page 510), and
ClefModifier (page 513).

Collision_engraver (page 413)
  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): NoteCollision (page 595).

Cue_clef_engraver (page 415)
  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.

forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly re-
  quested by the user.

forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

Custos_engraver (page 417)
Engrave custodes.
Properties (read)

forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): Custos (page 527).

Divisio_engraver (page 417)
Create divisiones: chant notation for points of breathing or caesura.
Music types accepted: caesura-event (page 50), fine-event (page 51), section-event (page 55), volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59),
Properties (read)

caesuraType (list)
An alist

((bar-line . bar-type)
(breath . breath-type)
(scripts . script-type...)
(underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

This engraver creates the following layout object(s): Divisio (page 528).

Dot_column_engraver (page 418)
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): DotColumn (page 531).
Figured_bass_engraver (page 421)
Make figured bass numbers.
Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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 exten-
  der lines.
useBassFigureExtenders (boolean)
  Whether to use extender lines for repeated bass figures.
This engraver creates the following layout object(s):
BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493),
BassFigureContinuation (page 494), and BassFigureLine (page 494).

Figured_bass_position_engraver (page 422)
Position figured bass alignments over notes.
This engraver creates the following layout object(s):
BassFigureAlignmentPositioning (page 492).

Fingering_column_engraver (page 422)
Find potentially colliding scripts and put them into a FingeringColumn object; that
will fix the collisions.
This engraver creates the following layout object(s): FingeringColumn (page 547).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)
fontSize (number)
  The relative size of all grobs in a context.

Grob_pq_engraver (page 426)
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.).
Instrument_name_engraver (page 427)
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 items (clef, key signature, etc.).

  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): InstrumentName (page 559).

Key_engraver (page 429)
Engrave a key signature.
Music types accepted: key-change-event (page 52),
Properties (read)

  createKeyOnClefChange (boolean)
  Print a key signature whenever the clef is changed.

  explicitKeySignatureVisibility (vector)
  ‘break-visibility’ function for explicit key changes. ‘\override
  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.

  forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly re-
  quested by the user.

  forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

  keyAlterationOrder (list)
  A list of pairs that defines in what order alterations should be printed.
  The format of an entry is (step . alter), where step is a number from
  0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact
  rationals for alterations in between, e.g., 1/2 for sharp.

  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): KeyCancellation (page 563), and KeySignature (page 565).

Ledger_line_engraver (page 431)
Create the spanner to draw ledger lines, and notices objects that need ledger lines.
This engraver creates the following layout object(s): LedgerLineSpanner (page 570).

Merge_mmrest_numbers_engraver (page 435)
Engraver to merge multi-measure rest numbers in multiple voices.
This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Non_musical_script_column_engraver (page 437)
Find potentially colliding non-musical scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Ottava_spanner_engraver (page 439)
Create a text spanner when the ottavation property changes.
Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Output_property_engraver (page 439)
Apply a procedure to any grob acknowledged.
Music types accepted: apply-output-event (page 48),

Piano_pedal_align_engraver (page 442)
Align piano pedal symbols and brackets.
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

Piano_pedal_engraver (page 442)
Engrave piano pedal symbols and brackets.
Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),
Properties (read)
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  pedalSostenutoStrings (list)
    See pedalSustainStrings.
  pedalSostenutoStyle (symbol)
    See pedalSustainStyle.
  pedalSustainStrings (list)
    A list of strings to print for sustain-pedal. Format is \texttt{(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): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

Pure_from_neighbor_engraver (page 443)
Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 444)
Handle collisions of rests.
Properties (read)

busyGros (list)
   A queue of (end-moment . grob) 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): RestCollision (page 612).

Script_row_engraver (page 445)
   Determine order in horizontal side position elements.
   This engraver creates the following layout object(s): ScriptRow (page 614).

Separating_line_group_engraver (page 445)
   Generate objects for computing spacing parameters.
   Properties (read)

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

   Properties (write)

   hasStaffSpacing (boolean)
      True if currentCommandColumn contains items that will affect spacing.

   This engraver creates the following layout object(s): StaffSpacing (page 632).

Skip_typesetting_engraver (page 446)
   Create a StaffEllipsis when skipTypesetting is used.
   Properties (read)

   skipTypesetting (boolean)
      If true, no typesetting is done, speeding up the interpretation phase.
      Useful for debugging large scores.

   This engraver creates the following layout object(s): StaffEllipsis (page 628).

Staff_collecting_engraver (page 448)
   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_highlight_engraver (page 449)
   Highlights music passages.
   Music types accepted: staff-highlight-event (page 56),
   Properties (read)

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

   This engraver creates the following layout object(s): StaffHighlight (page 631).
Staff_symbol_ engraver (page 449)
Create the constellation of five (default) staff lines.
Music types accepted: staff-span-event (page 57),
This engraver creates the following layout object(s): StaffSymbol (page 632).

2.1.40 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 (page 390).

This context creates the following layout object(s): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), DotColumn (page 531), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), Episema (page 542), FingerGlideSpanner (page 543), Fingering (page 545), Gissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), VaticanaLigature (page 667), and VoiceFollower (page 670).

This context sets the following properties:
• Set context property autoBeaming to #f.
• Set grob property bound-details.left.padding in Episema (page 542), to 0.05.
• Set grob property bound-details.right.padding in Episema (page 542), to 0.05.
• Set grob property style in NoteHead (page 596), to ’vaticana.punctum.
• Set grob property thickness in Episema (page 542), to 1.6.

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):

Arpeggio_ engraver (page 402)
Generate an Arpeggio symbol.
Music types accepted: arpeggio-event (page 49),
This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_ engraver (page 402)
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 2.2.141 [Stem_ engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.
Music types accepted: beam-forbid-event (page 49),
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 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): Beam (page 495).

Beam_engraver (page 407)
Handle Beam events by engraving beams. If omitted, then notes are printed with flags
instead of beams.

Music types accepted: beam-event (page 49),

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): Beam (page 495).

Bend_engraver (page 409)
Create fall spanners.

Music types accepted: bend-after-event (page 49),

Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current
timestep.

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

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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
Notate breath marks.
Music types accepted: breathing-event (page 50),
Properties (read)
  breathMarkType (symbol)
  The type of BreathingSign to create at \\
breath.
This engraver creates the following layout object(s): BreathingSign (page 502).

Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514),
and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
    quested by the user.
This engraver creates the following layout object(s): DoublePercentRepeat (page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner (page 538).

**Dynamic_engraver (page 420)**
Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56).

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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

**Episema_engraver (page 421)**
Create an *Editio Vaticana*-style episema line.

Music types accepted: episema-event (page 51),
This engraver creates the following layout object(s): Episema (page 542).

**Finger_glide_engraver (page 422)**
Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

**Fingering_engraver (page 422)**
Create fingering scripts.

Music types accepted: fingering-event (page 51),
This engraver creates the following layout object(s): Fingering (page 545).

**Font_size_engraver (page 422)**
Put fontSize into font-size grob property.

Properties (read)

- **fontSize (number)**
  The relative size of all grobs in a context.

**Forbid_line_break_engraver (page 423)**
Forbid line breaks when note heads are still playing at some point.
Chapter 2: Translation

Properties (read)

`busyGros (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, except if explicitly requested by the user.

**Glissando_engraver (page 424)**

Engrave glissandi.
Music types accepted: `glissando-event (page 51),`

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): Glissando (page 552).

**Grace_auto_beam_engraver (page 424)**

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: `beam-forbid-event (page 49),`

Properties (read)

`autoBeaming (boolean)`
If set to `true` then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).

**Grace_beam_engraver (page 425)**

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: `beam-event (page 49),`

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` 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): Beam (page 495).
Grace_engraver (page 425)
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.

Grob_pq_engraver (page 426)
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.).

Instrument_switch_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

instrumentCueName (markup)
The name to print if another instrument is to be taken.
This property is deprecated

This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Laissez_vibrer_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),
This engraver creates the following layout object(s): LaissezVibrerTie (page 569),
and LaissezVibrerTieColumn (page 570).

Multi_measure_rest_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow
and internalBarNumber to determine what number to print over the Section 3.1.88
[MultiMeasureRest], page 587.
Music types accepted: multi-measure-articulation-event (page 53),
multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)

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

internalBarNumber (integer)
Contains the current bar number. This property is used for internal time-
keeping, among others by the Accidental_engraver.
measureStartNow (boolean)
   True at the beginning of a measure.

restNumberThreshold (number)
   If a multimeasure rest has more measures than this, a number is printed.

This engraver creates the following layout object(s): MultiMeasureRest (page 587),
MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), and
MultiMeasureRestText (page 592).

New_fingering_engraver (page 437)
   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): Fingering (page 545), Script
(page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)
   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): VoiceFollower (page 670).

Note_heads_engraver (page 438)
   Generate note heads.

Music types accepted: note-event (page 54),

Properties (read)

   middleCPosition (number)
      The place of the middle C, measured in half staff-spaces. Usually deter-
      mined by looking at middleCClefPosition and middleCOffset.

   staffLineLayoutFunction (procedure)
      Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): NoteHead (page 596).

Note_spacing_engraver (page 439)
   Generate NoteSpacing, an object linking horizontal lines for use in spacing.
   This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
   Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),
Part_combine_engraver (page 441)
Part combine engraver for orchestral scores: Print markings 'a2', 'Solo', 'Solo II', and 'unisono'.
Music types accepted: note-event (page 54), and part-combine-event (page 54),
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?
  soloIIIText (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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
Make whole measure repeats.
Music types accepted: percent-event (page 54),
Properties (read)
  countPercentRepeats (boolean)
    If set, produce counters for percent repeats.
  currentCommandColumn (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature, etc.).
  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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
Print the bracketed note head after a note head with trill.
This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).
Repeat_tie_engraver (page 444)
Create repeat ties.
Music types accepted: repeat-tie-event (page 55),
This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
Engrave rests.
Music types accepted: rest-event (page 55),
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): Rest (page 611).

Rhythmic_column_engraver (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

Script_column_engraver (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

Script_engraver (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

Slash_repeat_engraver (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

Spanner_break_forbid_engraver (page 448)
Forbid breaks in certain spanners.

Text_engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engraver creates the following layout object(s): TextScript (page 650).

Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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): Tie (page 653), and
TieColumn (page 655).

**Trill_spanner_engraver** (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)

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

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): TrillSpanner (page 661).

**Tuplet_engraver** (page 455)
Catch tuplet events and generate appropriate bracket.
Music types accepted: tuplet-span-event (page 58),
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 signa-
tures, etc.) before the note.

This engraver creates the following layout object(s): TupletBracket (page 662),
and TupletNumber (page 664).

**Vaticana_ligature_engraver** (page 456)
Handle ligatures by glueing special ligature heads together.
Music types accepted: ligature-event (page 52), and pes-or-flexa-event
(page 55),

This engraver creates the following layout object(s): DotColumn (page 531), and
VaticanaLigature (page 667).
2.1.41 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): Arpeggio (page 483), Beam (page 495), BendAfter (page 497), BreathingSign (page 502), ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CombineTextScript (page 517), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), FingerGlideSpanner (page 543), Fingering (page 545), Flag (page 547), Glissando (page 552), Hairpin (page 555), InstrumentSwitch (page 560), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570), LigatureBracket (page 573), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NoteColumn (page 595), NoteHead (page 596), NoteSpacing (page 598), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RepeatSlash (page 609), RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611), Script (page 613), ScriptColumn (page 614), Slur (page 621), Stem (page 634), StemStub (page 636), StemTremolo (page 637), StringNumber (page 638), StrokeFinger (page 640), TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), and VoiceFollower (page 670).

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):

Arpeggio_engraver (page 402)
Generate an Arpeggio symbol.
Music types accepted: arpeggio-event (page 49),
This engraver creates the following layout object(s): Arpeggio (page 483).

Auto_beam_engraver (page 402)
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 2.2.141 [Stem_engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.
Music types accepted: beam-forbid-event (page 49),
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 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): Beam (page 495).

Beam_engraver (page 407)
   Handle Beam events by engraving beams. If omitted, then notes are printed with flags
   instead of beams.
   Music types accepted: beam-event (page 49),
   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): Beam (page 495).

Bend_engraver (page 409)
   Create fall spanners.
   Music types accepted: bend-after-event (page 49),
   Properties (read)
      currentBarLine (graphical (layout) object)
         Set to the BarLine that Bar_engraver has created in the current
timestep.
      currentCommandColumn (graphical (layout) object)
         Grob that is X-parent to all current breakable items (clef, key signature,
etc.).
      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): BendAfter (page 497).

Breathing_sign_engraver (page 410)
   Notate breath marks.
   Music types accepted: breathing-event (page 50),
   Properties (read)
      breathMarkType (symbol)
         The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).
Chord_tremolo_engraver (page 412)
Generate beams for tremolo repeats.
Music types accepted: tremolo-span-event (page 58),
This engraver creates the following layout object(s): Beam (page 495).

Cluster_spanner_engraver (page 413)
Engrave a cluster using Spanner notation.
Music types accepted: cluster-note-event (page 50),
This engraver creates the following layout object(s): ClusterSpanner (page 514),
and ClusterSpannerBeacon (page 515).

Dots_engraver (page 418)
Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.
This engraver creates the following layout object(s): Dots (page 531).

Double_percent_repeat_engraver (page 418)
Make double measure repeats.
Music types accepted: double-percent-event (page 50),
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, except if explicitly re-
    quested by the user.

This engraver creates the following layout object(s): DoublePercentRepeat
(page 532), and DoublePercentRepeatCounter (page 533).

Dynamic_align_engraver (page 419)
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): DynamicLineSpanner
(page 538).

Dynamic_engraver (page 420)
Create hairpins, dynamic texts and dynamic text spanners.
Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-
event (page 49), and span-dynamic-event (page 56),
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): DynamicText (page 539),
DynamicTextSpanner (page 541), and Hairpin (page 555).

Finger_glide_engraver (page 422)
Engraver to print a line between two Fingering grobs.
Music types accepted: note-event (page 54),
This engraver creates the following layout object(s): FingerGlideSpanner
(page 543).

Fingering_engraver (page 422)
Create fingering scripts.
Music types accepted: fingering-event (page 51),
This engraver creates the following layout object(s): Fingering (page 545).

Font_size_engraver (page 422)
Put fontSize into font-size grob property.
Properties (read)

fontSize (number)
The relative size of all grobs in a context.

Forbid_line_break_engraver (page 423)
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, except if explicitly re-
quested by the user.

Glissando_engraver (page 424)
Engrave glissandi.
Music types accepted: glissando-event (page 51),
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): `Glissando` (page 552).

**Grace_auto_beam_engraver** (page 424)
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: `beam-forbid-event` (page 49),

Properties (read)

`autoBeaming` (boolean)
If set to `true` then beams are generated automatically.

This engraver creates the following layout object(s): `Beam` (page 495).

**Grace_beam_engraver** (page 425)
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: `beam-event` (page 49),

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): `Beam` (page 495).

**Grace_engraver** (page 425)
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.

**Grob_pq_engraver** (page 426)
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)

busyGros (list)
A queue of \textit{(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.).

Instrument\_switch\_engraver (page 427)
Create a cue text for taking instrument.
This engraver is deprecated.
Properties (read)

instrument\_Cue\_Name (markup)
The name to print if another instrument is to be taken.
This property is deprecated

This engraver creates the following layout object(s): Instrument\_Switch (page 560).

Laissez\_vibrer\_engraver (page 430)
Create laissez vibrer items.
Music types accepted: laissez-vibrer-event (page 52),
This engraver creates the following layout object(s): Laissez\_Vibrer\_Tie (page 569),
and Laissez\_Vibrer\_Tie\_Column (page 570).

Ligature\_bracket\_engraver (page 431)
Handle Ligature\_events by engraving Ligature brackets.
Music types accepted: ligature-event (page 52),
This engraver creates the following layout object(s): Ligature\_Bracket (page 573).

Multi\_measure\_rest\_engraver (page 436)
Engrave multi-measure rests that are produced with ‘R’. It reads measure\_Start\_Now
and internal\_Bar\_Number to determine what number to print over the Section 3.1.88
[Multi\_Measure\_Rest], page 587.
Music types accepted: multi-measure-\_articulation\_event (page 53),
multi-measure-\_rest\_event (page 53), and multi-measure-\_text\_event (page 53),
Properties (read)

\begin{itemize}
  \item current\_Command\_Column (graphical (layout) object)
    Grob that is X-parent to all current breakable items (clef, key signature,
    etc.).
  \item internal\_Bar\_Number (integer)
    Contains the current bar number. This property is used for internal time-
    keeping, among others by the Accidental\_engraver.
  \item measure\_Start\_Now (boolean)
    True at the beginning of a measure.
  \item rest\_Number\_Threshold (number)
    If a multimeasure rest has more measures than this, a number is printed.
\end{itemize}

This engraver creates the following layout object(s): Multi\_Measure\_Rest (page 587),
Multi\_Measure\_Rest\_Number (page 589), Multi\_Measure\_Rest\_Script (page 590), and
Multi\_Measure\_Rest\_Text (page 592).

New\_fingering\_engraver (page 437)
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): Fingering (page 545), Script
(page 613), StringNumber (page 638), and StrokeFinger (page 640).

Note_head_line_engraver (page 438)
    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): VoiceFollower (page 670).

Note_heads_engraver (page 438)
    Generate note heads.
    Music types accepted: note-event (page 54),
Properties (read)

    middleCPosition (number)
        The place of the middle C, measured in half staff-spaces. Usually deter-
mined by looking at middleCClefPosition and middleCOffset.

staffLineLayoutFunction (procedure)
    Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): NoteHead (page 596).

Note_spacing_engraver (page 439)
    Generate NoteSpacing, an object linking horizontal lines for use in spacing.
    This engraver creates the following layout object(s): NoteSpacing (page 598).

Output_property_engraver (page 439)
    Apply a procedure to any grob acknowledged.
    Music types accepted: apply-output-event (page 48),

Part_combine_engraver (page 441)
    Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and
‘unisono’.
    Music types accepted: note-event (page 54), and part-combine-event (page 54),
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?

soloIIIText (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): CombineTextScript (page 517).

Percent_repeat_engraver (page 441)
   Make whole measure repeats.
   Music types accepted: percent-event (page 54),
   Properties (read)
      countPercentRepeats (boolean)
      If set, produce counters for percent repeats.
      currentCommandColumn (graphical (layout) object)
      Grob that is X-parent to all current breakable items (clef, key signature, etc.).
      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): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Phrasing_slur_engraver (page 441)
   Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.
   Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),
   This engraver creates the following layout object(s): PhrasingSlur (page 604).

Pitched_trill_engraver (page 443)
   Print the bracketed note head after a note head with trill.
   This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Repeat_tie_engraver (page 444)
   Create repeat ties.
   Music types accepted: repeat-tie-event (page 55),
   This engraver creates the following layout object(s): RepeatTie (page 610), and RepeatTieColumn (page 611).

Rest_engraver (page 444)
   Engrave rests.
   Music types accepted: rest-event (page 55),
   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): Rest (page 611).

**Rhythmic_column_engraver** (page 445)
Generate NoteColumn, an object that groups stems, note heads, and rests.
This engraver creates the following layout object(s): NoteColumn (page 595).

**Script_column_engraver** (page 445)
Find potentially colliding scripts and put them into a ScriptColumn object; that will fix the collisions.
This engraver creates the following layout object(s): ScriptColumn (page 614).

**Script_engraver** (page 445)
Handle note scripted articulations.
Music types accepted: articulation-event (page 49),
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): Script (page 613).

**Slash_repeat_engraver** (page 446)
Make beat repeats.
Music types accepted: repeat-slash-event (page 55),
This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

**Slur_engraver** (page 446)
Build slur grobs from slur events.
Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

**Spanner_break_forbid_engraver** (page 448)
Forbid breaks in certain spanners.

**Stem_engraver** (page 450)
Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.
Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),
Properties (read)
  
  currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.
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.

This engravor creates the following layout object(s): Flag (page 547), Stem
(page 634), StemStub (page 636), and StemTremolo (page 637).

Text_engraver (page 452)
Create text scripts.
Music types accepted: text-script-event (page 58),
This engravor creates the following layout object(s): TextScript (page 650).

Text_spanner_engraver (page 452)
Create text spanner from an event.
Music types accepted: text-span-event (page 58),
Properties (read)

currentMusicalColumn (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics,

This engravor creates the following layout object(s): TextSpanner (page 652).

Tie_engraver (page 452)
Generate ties between note heads of equal pitch.
Music types accepted: tie-event (page 58),
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 engravor creates the following layout object(s): Tie (page 653), and
TieColumn (page 655).

Trill_spanner_engraver (page 455)
Create trill spanners.
Music types accepted: trill-span-event (page 58),
Properties (read)

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

etc.).

currentMusicalColumn (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics,

etc.).
Chapter 2: Translation

This engraver creates the following layout object(s): TrillSpanner (page 661).

Tuplet_engraver (page 455)
- Catch tuplet events and generate appropriate bracket.
- Music types accepted: tuplet-span-event (page 58),
  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): TupletBracket (page 662), and TupletNumber (page 664).

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.
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 bar number. 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): Accidental (page 475),
AccidentalCautionary (page 476), AccidentalPlacement (page 477), and
AccidentalSuggestion (page 478).

Accidental_engraver is part of the following context(s) in \layout:
GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163),
KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff
(page 287), and VaticanaStaff (page 366).

2.2.2 Alteration_glyph_engraver
Set the glyph-name-alist of all grobs having the accidental-switch-interface to the value
of the context’s alterationGlyphs property, when defined.

Properties (read)

alterationGlyphs (list)
Alist mapping alterations to accidental glyphs. Alterations are given as exact num-
bers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

Alteration_glyph_engraver is part of the following context(s) in \layout: ChordGrid
(page 68), ChordNames (page 95), DrumStaff (page 108), GregorianTranscriptionStaff
2.2.3 Ambitus_ engraver

Create an ambitus.

Properties (read)

keyAlterations (list)
The current key signature. This is an list containing \( (\text{step} \cdot \text{alter}) \) or \( ((\text{octave} \cdot \text{step}) \cdot \text{alter}) \), where \( \text{step} \) is a number in the range 0 to 6 and \( \text{alter} \) a fraction, denoting alteration. For alterations, use symbols, e.g., \( \text{keyAlterations} = \#'((6 \cdot \text{FLAT})) \).

middleCClefPosition (number)
The position of the middle C, as determined only by the clef. This can be calculated by looking at clefPosition and clefGlyph.

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.

staffLineLayoutFunction (procedure)
Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): AccidentalPlacement (page 477), Ambitus (page 479), AmbitusAccidental (page 481), AmbitusLine (page 481), and AmbitusNoteHead (page 482).

Ambitus_ engraver is not part of any context

2.2.4 Arpeggio_ engraver

Generate an Arpeggio symbol.

Music types accepted: arpeggio-event (page 49),

This engraver creates the following layout object(s): Arpeggio (page 483).

Arpeggio_ engraver is part of the following context(s) in \layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.5 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 2.2.141 [Stem_ engraver], page 450, properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 49),
Chapter 2: Translation

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 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): Beam (page 495).

Auto_beam_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.6 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 items (clef, key signature, etc.).

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): VerticalAxisGroup (page 668).

Axis_group_engraver is part of the following context(s) in \layout: ChordGrid (page 68), ChordNames (page 95), DrumStaff (page 108), Dynamics (page 126), FiguredBass (page 132), FretBoards (page 133), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), OneStaff (page 230), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).
2.2.7 Balloon_engraver

Create balloon texts.

Music types accepted: annotate-output-event (page 48),
This engraver creates the following layout object(s): BalloonText (page 484).
Balloon_engraver is not part of any context

2.2.8 Bar_engraver

Create bar lines for various commands, including \bar.

If forbidBreakBetweenBarLines is true, allow line breaks at bar lines only.

Music types accepted: ad-hoc-jump-event (page 48), caesura-event (page 50),
coda-mark-event (page 50), dal-segno-event (page 50), fine-event (page 51),
section-event (page 55), segno-mark-event (page 56), and volta-span-event (page 59),

Properties (read)

caesuraType (list)

An alist

((bar-line . bar-type)
 (breath . breath-type)
 (scripts . script-type...)
 (underlying-bar-line . bar-type))

specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.
bar-line has higher priority than a measure bar line and underlying-bar-line
has lower priority than a measure bar line.

caesuraTypeTransform (procedure)

An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry
(articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

doubleRepeatBarType (string)
Bar line to insert where the end of one \repeat volta coincides with the start of another. The default is ‘\ldots’.
doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \repeat volta and the beginning of another. The default is ‘\ldots:.S.|:\’.
endRepeatBarType (string)
Bar line to insert at the end of a \repeat volta. The default is ‘:\ldots’.
endRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \repeat volta. The default is ‘:\ldots:S’.
fineBarType (string)
Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '|.'.

fineSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine. The default is '|.S'.

fineStartRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat volta. The default is '|.S.|:'.

forbidBreakBetweenBarLines (boolean)
If set to true, Bar_engraver forbids line breaks where there is no bar line.

measureBarType (string)
Bar line to insert at a measure boundary.

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '([command args...], but a command with no arguments may be abbreviated to a symbol; e.g., '(()(start-repeat))' may be given as '(start-repeat).

end-repeat
End a repeated section.

start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

sectionBarType (string)
Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '||'.

segnoBarType (string)
Bar line to insert at an in-staff segno. The default is 'S'.

segnoStyle (symbol)
A symbol that indicates how to print a segno: bar-line or mark.

startRepeatBarType (string)
Bar line to insert at the start of a \repeat volta. The default is '|:'.

startRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is 'S.|:'.

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '||'.

whichBar (string)
The current bar line type, or '()' if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.
Properties (write)

  currentBarLine (graphical (layout) object)
    Set to the BarLine that Bar_engraver has created in the current timestep.

  forbidBreak (boolean)
    If set to \#t, prevent a line break at this point, except if explicitly requested by the user.

This engraver creates the following layout object(s): BarLine (page 486).

Bar_engraver is part of the following context(s) in \layout: ChordGrid (page 68), DrumStaff (page 108), Dynamics (page 126), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.9 Bar_number_engraver

A bar number may be created at any bar line, subject to the barNumberVisibility callback. By default, 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 2.2.135 [Staff_collecting_engraver], page 448. This engraver usually creates BarNumber grobs, but when centerBarNumbers is true, it makes CenteredBarNumber grobs instead.

Properties (read)

  alternativeNumber (non-negative, exact integer)
    When set, the index of the current \alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple volte.

  alternativeNumberingStyle (symbol)
    The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

  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.

centerBarNumbers (boolean)
   Whether to center bar numbers in their measure instead of aligning them on the
   bar line.

currentBarNumber (integer)
   Contains the current bar number. This property is incremented at every bar line.

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

forbidBreak (boolean)
   If set to \#t, prevent a line break at this point, except if explicitly requested by the
   user.

forceBreak (boolean)
   Set to \#t when an event forcing a line break was heard.

measurePosition (moment)
   How much of the current measure have we had. This can be set manually to create
   incomplete measures.

stavesFound (list of grobs)
   A list of all staff-symbols found.

This engraver creates the following layout object(s): BarNumber (page 489), and
CenteredBarNumber (page 506).

Bar_number_engraver is part of the following context(s) in \layout: Score (page 263),
and StandaloneRhythmScore (page 301).

2.2.10 Beam_collision_engraver
Help beams avoid colliding with notes and clefs in other voices.

Beam_collision_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.11 Beam_engraver
Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead
of beams.

Music types accepted: beam-event (page 49),
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): Beam (page 495).

Beam_engraver is part of the following context(s) in \layout: CueVoice (page 97),
DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice
(page 189), MensuralVoice (page 216), NullVoice (page 228), PetrucciVoice (page 244),
StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and
Voice (page 390).

2.2.12 Beam_performer
Music types accepted: beam-event (page 49),

   Beam_performer is part of the following context(s) in \midi: ChordNames (page 95),
   CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153),
   KievanVoice (page 189), MensuralVoice (page 216), NullVoice (page 228), PetrucciVoice
   (page 244), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.13 Beat_engraver
This engraver is just a functionally identical copy of Section 2.2.14 [Beat_performer], page 409,
used for visualising its effects. You can also use it for showcasing the effects of the current
beatStructure.

   Music types accepted: articulation-event (page 49), and note-event (page 54),
   Properties (read)

   barExtraVelocity (integer)
      Extra MIDI velocity added by the ‘Beat_performer’ at the start of each measure.
   baseMoment (moment)
      Smallest unit of time that will stand on its own as a subdivided section.
   beatExtraVelocity (integer)
      Extra MIDI velocity added by the ‘Beat_performer’ at the start of each beat.
   beatStructure (list)
      List of baseMoments that are combined to make beats.
   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 (boolean)
      Keep administration of measure length, position, bar number, etc.? Switch off for
cadenzas.

Beat_engraver is not part of any context
2.2.14 Beat_performer

This performer is intended for instantiation in ‘Voice’-like contexts. The context variable beatExtraVelocity is used for adding extra MIDI velocity at each beat (default 15) in accordance with beatStructure and an additional barExtraVelocity (default 10) at the start of each bar.

This is done by adding corresponding \accent and \marcato events when such note events are encountered.

Off-beat manual use of \accent or \marcato causes autogeneration of the next on-beat accent to be skipped.

Music types accepted: articulation-event (page 49), and note-event (page 54).

Properties (read)

barExtraVelocity (integer)
Extra MIDI velocity added by the ‘Beat_performer’ at the start of each measure.

baseMoment (moment)
Smallest unit of time that will stand on its own as a subdivided section.

beatExtraVelocity (integer)
Extra MIDI velocity added by the ‘Beat_performer’ at the start of each beat.

beatStructure (list)
List of baseMoments that are combined to make beats.

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 (boolean)
Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

Beat_performer is not part of any context

2.2.15 Bend_engraver

Create fall spanners.

Music types accepted: bend-after-event (page 49).

Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

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

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): BendAfter (page 497).

Bend_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievianVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.16 Bend_spanner_engraver

Engraver to print a BendSpanner.

Music types accepted: bend-span-event (page 49), note-event (page 54), and string-number-event (page 57).

Properties (read)

stringFretFingerList (list)
A list containing three entries. In TabVoice and FretBoards they determine the string, fret and finger to use

supportNonIntegerFret (boolean)
If set in Score the TabStaff will print micro-tones as ‘2½’

Properties (write)

stringFretFingerList (list)
A list containing three entries. In TabVoice and FretBoards they determine the string, fret and finger to use

supportNonIntegerFret (boolean)
If set in Score the TabStaff will print micro-tones as ‘2½’

This engraver creates the following layout object(s): BendSpanner (page 498).

Bend_spanner_engraver is part of the following context(s) in \layout: TabVoice (page 352).

2.2.17 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): BreakAlignGroup (page 500), BreakAlignment (page 501), and LeftEdge (page 571).

Break_align_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.18 Breathing_sign_engraver

Notate breath marks.

Music types accepted: breathing-event (page 50),

Properties (read)

breathMarkType (symbol)
The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 502).

Breathing_sign_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.19 Caesura_engraver

Notate a short break in sound that does not shorten the previous note.

Depending on the result of passing the value of caesuraType through caesuraTypeTransform, this engraver may create a BreathingSign with CaesuraScript grobs aligned to it, or it may create CaesuraScript grobs and align them to a BarLine.
If this engraver observes a BarLine, it calls caesuraTypeTransform again with the new information, and if necessary, recreates its grobs.

Music types accepted: caesura-event (page 50),

Properties (read)

breathMarkDefinitions (list)
   The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

caesuraType (list)
   An alist
   
   ((bar-line . bar-type)
    (breath . breath-type)
    (scripts . script-type...)
    (underlying-bar-line . bar-type))

   specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.
   bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

caesuraTypeTransform (procedure)
   An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
   The first argument is the context.
   The second argument is the value of caesuraType with an additional entry
   (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
   The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

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): BreathingSign (page 502), and CaesuraScript (page 504).

Caesura_engraver is part of the following context(s) in \layout: DrumStaff (page 108), KievanStaff (page 176), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), and TabStaff (page 341).

2.2.20 Centered_bar_number_align_engraver

Group measure-centered bar numbers in a CenteredBarNumberLineSpanner so they end up on the same vertical position.

Properties (read)

currentCommandColumn (graphical (layout) object)
   Grob that is X-parent to all current breakable items (clef, key signature, etc.).
This engraver creates the following layout object(s): CenteredBarNumberLineSpanner (page 507).

Centered_bar_number_align_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

### 2.2.21 Chord_name_engraver

Read currentChordText to create chord names.

Properties (read)

- chordChanges (boolean)
  - Only show changes in chords scheme?

- currentChordCause (stream event)
  - Event cause of the chord that should be created in this time step (if any).

- currentChordText (markup)
  - In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

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

Properties (write)

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

This engraver creates the following layout object(s): ChordName (page 508).

Chord_name_engraver is part of the following context(s) in \layout: ChordNames (page 95).

### 2.2.22 Chord_square_engraver

Engrave chord squares in chord grids.

Properties (read)

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

This engraver creates the following layout object(s): ChordSquare (page 509).

Chord_square_engraver is part of the following context(s) in \layout: ChordGrid (page 68).

### 2.2.23 Chord_tremolo_engraver

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 58).

This engraver creates the following layout object(s): Beam (page 495).

Chord_tremolo_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.24 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.

- forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly requested by the
  user.

- forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

- 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): Clef (page 510), and ClefModifier
(page 513).

Clef_engraver is part of the following context(s) in \layout: DrumStaff (page 108),
GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163),
KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff
(page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.25 Cluster_spanner_engraver

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 50),

This engraver creates the following layout object(s): ClusterSpanner (page 514), and
ClusterSpannerBeacon (page 515).

Cluster_spanner_engraver is part of the following context(s) in \layout: CueVoice
(page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189),
MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331),
TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.26 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): NoteCollision (page 595).

Collision_engraver is part of the following context(s) in \layout: DrumStaff
(page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff
(page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

### 2.2.27 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:** note-event (page 54),

**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): `NoteHead` (page 596), Tie (page 653), and TieColumn (page 655).

Completion_heads_engraver is not part of any context.

### 2.2.28 Completion_rest_engraver

This engraver replaces Rest_engraver. It plays some trickery to break long rests into the next measure.

**Music types accepted:** rest-event (page 55),

**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): Rest (page 611).
Completion_rest_engraver is not part of any context.

2.2.29 Concurrent_hairpin_engraver
Collect concurrent hairpins.

Concurrent_hairpin_engraver is part of the following context(s) in \layout:
ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.30 Control_track_performer
Properties (read)

midiSkipOffset (moment)
    This is the accrued MIDI offset to account for time skipped via skipTypesetting.

Control_track_performer is part of the following context(s) in \midi: Score (page 263).

2.2.31 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.
forbidBreak (boolean)
    If set to #t, prevent a line break at this point, except if explicitly requested by the user.

forceBreak (boolean)
    Set to #t when an event forcing a line break was heard.

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): ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

Cue_clef_engraver is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.32 Current_chord_text_engraver

Catch note and rest events and generate the appropriate chord text using chordNameFunction. Actually creating a chord name grob is left to other engravers.

Music types accepted: general-rest-event (page 51), and note-event (page 54),

Properties (read)

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.

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)

currentChordCause (stream event)
    Event cause of the chord that should be created in this time step (if any).

currentChordText (markup)
    In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

Current_chord_text_engraver is part of the following context(s) in \layout: ChordGrid (page 68), and ChordNames (page 95).
2.2.33 Custos_engraver

Engrave custodes.

Properties (read)

- forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly requested by the user.

- forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): Custos (page 527).

Custos_engraver is part of the following context(s) in layout: MensuralStaff (page 202), PetrucciStaff (page 231), and VaticanaStaff (page 366).

2.2.34 Divisio_engraver

Create divisiones: chant notation for points of breathing or caesura.

Music types accepted: caesura-event (page 50), fine-event (page 51), section-event (page 55), volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59).

Properties (read)

- caesuraType (list)
  An alist
  
  ```
  ((bar-line . bar-type)
   (breath . breath-type)
   (scripts . script-type...)
   (underlying-bar-line . bar-type))
  ```

  specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.
  
  bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.

- caesuraTypeTransform (procedure)
  An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
  
  The first argument is the context.
  
  The second argument is the value of caesuraType with an additional entry (articularions . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
  
  The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

This engraver creates the following layout object(s): Divisio (page 528).

Divisio_engraver is part of the following context(s) in layout: GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), MensuralStaff (page 202), and VaticanaStaff (page 366).
2.2.35 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): DotColumn (page 531).

Dot_column_engraver is part of the following context(s) in layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.36 Dots_engraver

Create Section 3.1.43 [Dots], page 531, objects for Section 3.2.118 [rhythmic-head-interface], page 734s.

This engraver creates the following layout object(s): Dots (page 531).

Dots_engraver is part of the following context(s) in layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.37 Double_percent_repeat_engraver

Make double measure repeats.

Music types accepted: double-percent-event (page 50),

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, except if explicitly requested by the
    user.

This engraver creates the following layout object(s): DoublePercentRepeat (page 532),
and DoublePercentRepeatCounter (page 533).

Double_percent_repeat_engraver is part of the following context(s) in layout: ChordGrid (page 68), CueVoice (page 97), DrumVoice (page 117),
GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.38 Drum_note_performer

Play drum notes.

Music types accepted: articulation-event (page 49), note-event (page 54), and tie-event (page 58),

Drum_note_performer is part of the following context(s) in midi: DrumVoice (page 117).
2.2.39 Drum_notes_engraver

Generate drum note heads.

Music types accepted: note-event (page 54),

Properties (read)

drumStyleTable (hash table)
A hash table which maps drums to layout settings. Predefined values:
'drums-style', 'agostini-drums-style', 'weinberg-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): NoteHead (page 596), and Script (page 613).

Drum_notes_engraver is part of the following context(s) in \layout: DrumVoice (page 117).

2.2.40 Duration_line_engraver

Engraver to print a line representing the duration of a rhythmic event like NoteHead, NoteColumn or Rest.

Music types accepted: duration-line-event (page 51),

Properties (read)

currentCommandColumn (graphical (layout) object)
Grob that is X-parent to all current breakable items (clef, key signature, etc.).
currentMusicalColumn (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).
edAtSkip (boolean)
End DurationLine grob on skip-event
startAtNoteColumn (boolean)
Start DurationLine grob at entire NoteColumn.
startAtSkip (boolean)
Start DurationLine grob at skip-event.

This engraver creates the following layout object(s): DurationLine (page 536).

Duration_line_engraver is not part of any context.

2.2.41 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): DynamicLineSpanner (page 538).

Dynamic_align_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), Dynamics (page 126), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.42 Dynamic_engraver

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 48), break-dynamic-span-event (page 49), and span-dynamic-event (page 56).

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): DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

Dynamic_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), Dynamics (page 126), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.43 Dynamic_performer

Music types accepted: absolute-dynamic-event (page 48), crescendo-event (page 50), and decrescendo-event (page 50).

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 \((\text{min} \ . \ \text{max})\) 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 part of the following context(s) in \midi: ChordNames (page 95), CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.44 Episema_engraver

Create an Editio Vaticana-style episema line.

Music types accepted: episema-event (page 51),
This engraver creates the following layout object(s): Episema (page 542).
Episema_engraver is part of the following context(s) in \layout:
GregorianTranscriptionVoice (page 153), and VaticanaVoice (page 380).

2.2.45 Extender_engraver

Create lyric extenders.

Music types accepted: completize-extender-event (page 50), and extender-event (page 51),
Properties (read)

    extendersOverRests (boolean)
    Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): LyricExtender (page 574).
Extender_engraver is part of the following context(s) in \layout:
GregorianTranscriptionLyrics (page 137), Lyrics (page 199), and VaticanaLyrics (page 363).

2.2.46 Figured_bass_engraver

Make figured bass numbers.

Music types accepted: bass-figure-event (page 49), and rest-event (page 55),
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): BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureBracket (page 493), BassFigureContinuation (page 494), and BassFigureLine (page 494).
Figured_bass_engraver is part of the following context(s) in \layout: DrumStaff (page 108), FiguredBass (page 132), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).
2.2.47 Figured_bass_position_engraver
Position figured bass alignments over notes.

This engraver creates the following layout object(s): BassFigureAlignmentPositioning (page 492).

Figured_bass_position_engraver is part of the following context(s) in \layout:
DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.48 Finger_glide_engraver
Engraver to print a line between two Fingering grobs.

Music types accepted: note-event (page 54),

This engraver creates the following layout object(s): FingerGlideSpanner (page 543).

Finger_glide_engraver is part of the following context(s) in \layout: DrumVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.49 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): FingeringColumn (page 547).

Fingering_column_engraver is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.50 Fingering_engraver
Create fingering scripts.

Music types accepted: fingering-event (page 51),

This engraver creates the following layout object(s): Fingering (page 545).

Fingering_engraver is part of the following context(s) in \layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), VaticanaVoice (page 380), and Voice (page 390).

2.2.51 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) in \layout: CueVoice (page 97), DrumStaff (page 108), DrumVoice (page 117), Dynamics (page 126), FretBoards (page 133), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), GregorianTranscriptionVoice (page 153), InternalGregorianStaff (page 163), KievanStaff (page 176), KievanVoice (page 189), Lyrics (page 199), MensuralStaff (page 202), MensuralVoice (page 216), PetrucciStaff (page 231), PetrucciVoice (page 244), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff
2.2.52 Footnote_engraver

Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 548).

Footnote_engraver is part of the following context(s) in `\layout`: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.53 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, except if explicitly requested by the user.

Forbid_line_break_engraver is part of the following context(s) in `\layout`: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.54 Fretboard_engraver

Generate fret diagram from one or more events of type NoteEvent.

Music types accepted: fingering-event (page 51), note-event (page 54), and string-number-event (page 57),

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.
Chapter 2: Translation

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): FretBoard (page 550).
Fretboard_engraver is part of the following context(s) in layout: FretBoards (page 133).

2.2.55 Glissando_engraver
Engrave glissandi.

Music types accepted: glissando-event (page 51),

Properties (read)

  glissandoMap (list)
  A map in the form of '((source1 . target1) (source2 . target2) (source3 . target3)) 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): Glissando (page 552).
Glissando_engraver is part of the following context(s) in layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.56 Grace_auto_beam_engraver
Generates one autobeam group across an entire grace phrase. As usual, any manual beaming or \noBeam will block autobeam, just like setting the context property 'autoBeaming' to #f.

Music types accepted: beam-forbid-event (page 49),

Properties (read)

  autoBeaming (boolean)
  If set to true then beams are generated automatically.

This engraver creates the following layout object(s): Beam (page 495).
Grace_auto_beam_engraver is part of the following context(s) in layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.57 **Grace_beam_engraver**

Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engrav es beams when we are at grace points in time.

Music types accepted: beam-event (page 49),

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): Beam (page 495).

Grace_beam_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.58 **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) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.59 **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): GraceSpacing (page 553).

Grace_spacing_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.60 **Grid_chord_name_engraver**

Read currentChordText to create chord names adapted for typesetting within a chord grid.

Properties (read)

- **currentChordCause** (stream event)
  Event cause of the chord that should be created in this time step (if any).
currentChordText (markup)
In contexts printing chord names, this is at any point of time the markup that will
be put in the chord name.

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

This engraver creates the following layout object(s): GridChordName (page 553).
Grid_chord_name_engraver is part of the following context(s) in \layout: ChordGrid
(page 68).

2.2.61 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): GridLine (page 554).
Grid_line_span_engraver is not part of any context

2.2.62 Grid_point_engraver
Generate grid points.
Properties (read)
  gridInterval (moment)
  Interval for which to generate GridPoints.

This engraver creates the following layout object(s): GridPoint (page 555).
Grid_point_engraver is not part of any context

2.2.63 Grob_pq_engraver
Administrate when certain grobs (e.g., note heads) stop playing.
Properties (read)
  busyGrobs (list)
  A queue of \(\text{end-moment . grob}\) 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 \(\text{end-moment . grob}\) 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) in \layout: CueVoice
(page 97), DrumStaff (page 108), DrumVoice (page 117), GregorianTranscriptionStaff
(page 140), GregorianTranscriptionVoice (page 153), InternalGregorianStaff
(page 163), KievanStaff (page 176), KievanVoice (page 189), MensuralStaff
(page 202), MensuralVoice (page 216), NullVoice (page 228), PetrucciStaff (page 231),
PetrucciVoice (page 244), Staff (page 287), StandaloneRhythmVoice (page 331), TabStaff
(page 341), TabVoice (page 352), VaticanaStaff (page 366), VaticanaVoice (page 380), and
Voice (page 390).
2.2.64 **Horizontal_bracket_engraver**

Create horizontal brackets over notes for musical analysis purposes.

- **Music types accepted:** note-grouping-event (page 54),
- This engraver creates the following layout object(s): HorizontalBracket (page 557), and HorizontalBracketText (page 558).
- Horizontal_bracket_engraver is not part of any context

2.2.65 **Hyphen_engraver**

Create lyric hyphens, vowel transitions and distance constraints between words.

- **Music types accepted:** hyphen-event (page 52), and vowel-transition-event (page 59),
- This engraver creates the following layout object(s): LyricHyphen (page 575), LyricSpace (page 578), and VowelTransition (page 673).
- Hyphen_engraver is part of the following context(s) in \layout:
  - GregorianTranscriptionLyrics (page 137), Lyrics (page 199), and VaticanaLyrics (page 363).

2.2.66 **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 items (clef, key signature, etc.).
  - 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):** InstrumentName (page 559).
- **Instrument_name_engraver is part of the following context(s) in \layout:**
  - ChoirStaff (page 66), DrumStaff (page 108), FretBoards (page 133), GrandStaff (page 135), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievianStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), PetrucciStaff (page 231), PianoStaff (page 255), RhythmicStaff (page 257), Staff (page 287), StaffGroup (page 299), standaloneRhythmStaff (page 325), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

2.2.67 **Instrument_switch_engraver**

Create a cue text for taking instrument.

- This engraver is deprecated.

- **Properties (read)**
  - instrumentCueName (markup)
    - The name to print if another instrument is to be taken.
    - This property is deprecated
This engraver creates the following layout object(s): InstrumentSwitch (page 560).

Instrument_switch_engraver is part of the following context(s) in layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.68 Jump_engraver

This engraver creates instructions such as D.C. and Fine, placing them vertically outside the set of staves given in the stavesFound context property.

If Jump_engraver is added or moved to another context, Staff_collecting_engraver (page 448), also needs to be there so that marks appear at the intended Y location.

Music types accepted: ad-hoc-jump-event (page 48), dal-segno-event (page 50), and fine-event (page 51).

Properties (read)

- **codaMarkCount** (non-negative, exact integer)
  Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

- **codaMarkFormatter** (procedure)
  A procedure that creates a coda mark (which in conventional D.S. al Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.

- **dalSegnoTextFormatter** (procedure)
  Format a jump instruction such as D.S.
  The first argument is the context.
  The second argument is the number of times the instruction is performed.
  The third argument is a list of three markups: start-markup, end-markup, and next-markup.
  If start-markup is #f, the form is da capo; otherwise the form is dal segno and start-markup is the sign at the start of the repeated section.
  If end-markup is not #f, it is either the sign at the end of the main body of the repeat, or it is a Fine instruction. When it is a Fine instruction, next-markup is #f.
  If next-markup is not #f, it is the mark to be jumped to after performing the body of the repeat, e.g., Coda.

- **finalFineTextVisibility** (boolean)
  Whether \fine at the written end of the music should create a Fine instruction.

- **fineText** (markup)
  The text to print at \fine.

- **segnoMarkCount** (non-negative, exact integer)
  Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

- **segnoMarkFormatter** (procedure)
  A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.
stavesFound (list of grobs)
A list of all staff-symbols found.

This engraver creates the following layout object(s): JumpScript (page 561).
Jump_script is part of the following context(s) in \layout: ChordGridScore (page 73),
Score (page 263), and StandaloneRhythmScore (page 301).

2.2.69 Keep_alive_together_engraver
This engraver collects all Hara_kiri_group_spanners 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) in \layout:
PianoStaff (page 255).

2.2.70 Key_engraver
Engrave a key signature.
Music types accepted: key-change-event (page 52),
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.
forbidBreak (boolean)
If set to #t, prevent a line break at this point, except if explicitly requested by the
user.
forceBreak (boolean)
Set to #t when an event forcing a line break was heard.
keyAlterationOrder (list)
A list of pairs that defines in what order alterations should be printed. The format
of an entry is (step . alter), where step is a number from 0 to 6 and alter from
-1 (double flat) to 1 (double sharp), with exact rationals for alterations in between,
e.g., 1/2 for sharp.
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 calcu-
lated 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): KeyCancellation (page 563), and KeySignature (page 565).

Key_performer is part of the following context(s) in \layout:
GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163),
KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), and VaticanaStaff (page 366).

2.2.71 Key_performer
Music types accepted: key-change-event (page 52),

Properties (read)

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 quotes.

Key_performer is part of the following context(s) in \midi: DrumStaff (page 108),
GregorianTranscriptionStaff (page 140), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287),
TabStaff (page 341), and VaticanaStaff (page 366).

2.2.72 Kievan_ligature_engraver
Handle Kievan_ligature_events by glueing Kievan heads together.

Music types accepted: ligature-event (page 52),

This engraver creates the following layout object(s): KievanLigature (page 568).

Kievan_ligature_engraver is part of the following context(s) in \layout: KievanVoice (page 189).

2.2.73 Laissez_vibrer_engraver
Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 52),

This engraver creates the following layout object(s): LaissezVibrerTie (page 569), and LaissezVibrerTieColumn (page 570).

Laissez_vibrer_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), standaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.74 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): LedgerLineSpanner (page 570).

Ledger_line_engraver is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.75 Ligature_bracket_engraver

Handle Ligature_events by engraving Ligature brackets.

Music types accepted: ligature-event (page 52),

This engraver creates the following layout object(s): LigatureBracket (page 573).

Ligature_bracket_engraver is part of the following context(s) in \layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), StandaloneRhythmVoice (page 331), TabVoice (page 352), and Voice (page 390).

2.2.76 Lyric_engraver

Engrave text for lyrics.

Music types accepted: lyric-event (page 52),

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): LyricText (page 579).

Lyric_engraver is part of the following context(s) in \layout:
GregorianTranscriptionLyrics (page 137), Lyrics (page 199), and VaticanaLyrics (page 363).

2.2.77 Lyric_performer

Music types accepted: lyric-event (page 52),

Lyric_performer is part of the following context(s) in \midi:
GregorianTranscriptionLyrics (page 137), and Lyrics (page 199).

2.2.78 Lyric_repeat_count_engraver

Create repeat counts within lyrics for modern transcriptions of Gregorian chant.

Music types accepted: volta-repeat-end-event (page 59),

Properties (read)

  lyricRepeatCountFormatter (procedure)
  A procedure taking as arguments the context and the numeric repeat count. It should return the formatted repeat count as markup. If it does not return markup, no grob is created.
This engraver creates the following layout object(s): LyricRepeatCount (page 576).

Lyric_repeat_count_engraver is part of the following context(s) in \layout:
GregorianTranscriptionLyrics (page 137).

2.2.79 Mark_engraver

This engraver creates rehearsal marks, segno and coda marks, and section labels.

Mark_engraver creates marks, formats them, and places them vertically outside the set of staves given in the stavesFound context property.

If Mark_engraver is added or moved to another context, Staff_collecting_engraver (page 448), also needs to be there so that marks appear at the intended Y location.

By default, Mark_engravers in multiple contexts create a common sequence of marks chosen by the Score-level Mark_tracking_translator (page 433). If independent sequences are desired, multiple Mark_tracking_translators must be used.

Properties (read)

- codaMarkFormatter (procedure)
  A procedure that creates a coda mark (which in conventional D.S. al Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.

- currentPerformanceMarkEvent (stream event)
  The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

- currentRehearsalMarkEvent (stream event)
  The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

- rehearsalMarkFormatter (procedure)
  A procedure taking as arguments the context and the sequence number of the rehearsal mark. It should return the formatted mark as a markup object.

- segnoMarkFormatter (procedure)
  A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.

- stavesFound (list of grobs)
  A list of all staff-symbols found.

This engraver creates the following layout object(s): CodaMark (page 515), RehearsalMark (page 607), SectionLabel (page 615), and SegnoMark (page 616).

Mark_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.80 Mark_performer

This performer emits MIDI markers for rehearsal marks, segno and coda marks, and section labels. The MIDI markers are derived from markup that is generated as in the Mark_engraver.

Properties (read)

- currentPerformanceMarkEvent (stream event)
  The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.
currentRehearsalMarkEvent (stream event)
  The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for
  engraving by Mark_engraver.

Mark_performer is part of the following context(s) in \midi: Score (page 263).

\subsection{Mark_tracking_translator}
This translator chooses which marks Mark_engraver should engrave.

\begin{itemize}
  \item Music types accepted: ad-hoc-mark-event (page 48), coda-mark-event (page 50),
  rehearsal-mark-event (page 55), section-label-event (page 55), and segno-mark-event
  (page 56),
\end{itemize}

\begin{itemize}
  \item Properties (read)
    \begin{itemize}
      \item codaMarkCount (non-negative, exact integer)
        Updated at the end of each timestep in which a coda mark appears: not set during
        the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from
        the second to the third, etc.
      \item rehearsalMark (integer)
        The next rehearsal mark to print.
      \item segnoMarkCount (non-negative, exact integer)
        Updated at the end of each timestep in which a segno appears: not set during
        the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from
        the second to the third segno, etc.
    \end{itemize}
  \item Properties (write)
    \begin{itemize}
      \item codaMarkCount (non-negative, exact integer)
        Updated at the end of each timestep in which a coda mark appears: not set during
        the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from
        the second to the third, etc.
      \item currentPerformanceMarkEvent (stream event)
        The coda, section, or segno mark event selected by Mark_tracking_translator for
        engraving by Mark_engraver.
      \item currentRehearsalMarkEvent (stream event)
        The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for
        engraving by Mark_engraver.
      \item rehearsalMark (integer)
        The next rehearsal mark to print.
      \item segnoMarkCount (non-negative, exact integer)
        Updated at the end of each timestep in which a segno appears: not set during
        the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from
        the second to the third segno, etc.
    \end{itemize}
\end{itemize}

Mark_tracking_translator is part of the following context(s) in \layout:
ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301); in
\midi: Score (page 263).

\subsection{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: \textit{measure-counter-event} (page 52),

Properties (read)

\begin{description}
\item[currentBarNumber] (integer) 
  Contains the current bar number. This property is incremented at every bar line.
\item[currentCommandColumn] (graphical (layout) object) 
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
\item[measurePosition] (moment) 
  How much of the current measure have we had. This can be set manually to create incomplete measures.
\end{description}

This engraver creates the following layout object(s): MeasureCounter (page 580).

\textit{Measure_counter_engraver} is not part of any context

\subsection{2.2.83 Measure\textunderscore grouping\textunderscore engraver}

Create MeasureGrouping to indicate beat subdivision.

Properties (read)

\begin{description}
\item[baseMoment] (moment) 
  Smallest unit of time that will stand on its own as a subdivided section.
\item[beatStructure] (list) 
  List of baseMoments that are combined to make beats.
\item[currentMusicalColumn] (graphical (layout) object) 
  Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).
\item[measurePosition] (moment) 
  How much of the current measure have we had. This can be set manually to create incomplete measures.
\end{description}

This engraver creates the following layout object(s): MeasureGrouping (page 582).

\textit{Measure\_grouping\_engraver} is not part of any context

\subsection{2.2.84 Measure\textunderscore spanner\textunderscore engraver}

This engraver creates spanners bounded by the columns that start and end measures in response to \texttt{\startMeasureSpanner} and \texttt{\stopMeasureSpanner}.

Music types accepted: \textit{measure-spanner-event} (page 53),

Properties (read)

\begin{description}
\item[currentCommandColumn] (graphical (layout) object) 
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
\item[measurePosition] (moment) 
  How much of the current measure have we had. This can be set manually to create incomplete measures.
\end{description}

This engraver creates the following layout object(s): MeasureSpanner (page 583).

\textit{Measure\_spanner\_engraver} is not part of any context
2.2.85 Melody_engraver
Create information for context dependent typesetting decisions.

Properties (read)

- currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_engraver has created in the current timestep.

- suspendMelodyDecisions (boolean)
  When using the Melody_engraver, stop changing orientation of stems based on the melody when this is set to true.

This engraver creates the following layout object(s): MelodyItem (page 585).

Melody_engraver is not part of any context

2.2.86 Mensural_ligature_engraver
Handle Mensural_ligature_events by glueing special ligature heads together.

Music types accepted: ligature-event (page 52),

This engraver creates the following layout object(s): MensuralLigature (page 585).

Mensural_ligature_engraver is part of the following context(s) in \layout:
MensuralVoice (page 216), and PetrucciVoice (page 244).

2.2.87 Merge_mmrest_numbers_engraver
Engraver to merge multi-measure rest numbers in multiple voices.

This works by gathering all multi-measure rest numbers at a time step. If they all have the same text and there are at least two only the first one is retained and the others are hidden.

Merge_mmrest_numbers_engraver is part of the following context(s) in \layout:
DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.88 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 all are 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

2.2.89 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 2.2.135 \[Staff_collecting_engraver\], page 448.

Music types accepted: tempo-change-event (page 58),

Properties (read)

- currentCommandColumn (graphical (layout) object)
  Grob that is X-parent to all current breakable items (clef, key signature, etc.).
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): MetronomeMark (page 585).

Metronome_mark_engraver is part of the following context(s) in layout:
ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.90 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 part of the following context(s) in midi:
DrumStaff (page 108), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.91 Multi_measure_rest_engraver
Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the Section 3.1.88 [MultiMeasureRest], page 587.

Music types accepted: multi-measure-articulation-event (page 53), multi-measure-rest-event (page 53), and multi-measure-text-event (page 53),
Properties (read)

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

internalBarNumber (integer)
  Contains the current bar number. This property is used for internal timekeeping,
  among others by the Accidental_engraver.

measureStartNow (boolean)
  True at the beginning of a measure.

restNumberThreshold (number)
  If a multimeasure rest has more measures than this, a number is printed.
```

This engraver creates the following layout object(s): `MultiMeasureRest` (page 587), `MultiMeasureRestNumber` (page 589), `MultiMeasureRestScript` (page 590), and `MultiMeasureRestText` (page 592).

`Multi_measure_rest_engraver` is part of the following context(s) in `\layout`:
- `CueVoice` (page 97), `DrumVoice` (page 117), `GregorianTranscriptionVoice` (page 153), `KievanVoice` (page 189), `MensuralVoice` (page 216), `PetrucciVoice` (page 244), `StandaloneRhythmVoice` (page 331), `TabVoice` (page 352), `VaticanaVoice` (page 380), and `Voice` (page 390).

### 2.2.92 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)

```plaintext
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): `Fingering` (page 545), `Script` (page 613), `StringNumber` (page 638), and `StrokeFinger` (page 640).

`New_fingering_engraver` is part of the following context(s) in `\layout`:
- `CueVoice` (page 97), `GregorianTranscriptionVoice` (page 153), `KievanVoice` (page 189), `MensuralVoice` (page 216), `PetrucciVoice` (page 244), `StandaloneRhythmVoice` (page 331), `VaticanaVoice` (page 380), and `Voice` (page 390).

### 2.2.93 Non_musical_script_column_engraver

Find potentially colliding non-musical scripts and put them into a `ScriptColumn` object; that will fix the collisions.

This engraver creates the following layout object(s): `ScriptColumn` (page 614).

`Non_musical_script_column_engraver` is part of the following context(s) in `\layout`:
- `DrumStaff` (page 108), `GregorianTranscriptionStaff` (page 140), `InternalGregorianStaff` (page 163), `KievanStaff` (page 176), `MensuralStaff` (page 202), `PetrucciStaff` (page 231), `Staff` (page 287), `TabStaff` (page 341), and `VaticanaStaff` (page 366).
2.2.94 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): VoiceFollower (page 670).

Note_head_line_engraver is part of the following context(s) in layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.95 Note_heads_engraver

Generate note heads.

Music types accepted: note-event (page 54),

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): NoteHead (page 596).

Note_heads_engraver is part of the following context(s) in layout: CueVoice (page 97), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), NullVoice (page 228), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), VaticanaVoice (page 380), and Voice (page 390).

2.2.96 Note_name_engraver

Print pitches as words.

Music types accepted: note-event (page 54),

Properties (read)

noteNameFunction (procedure)

Function used to convert pitches into strings and markups.

noteNameSeparator (string)

String used to separate simultaneous NoteName objects.

printAccidentalNames (boolean or symbol)

Print accidentals in the NoteNames context.

printNotesLanguage (string)

Use a specific language in the NoteNames context.

printOctaveNames (boolean or symbol)

Print octave marks in the NoteNames context.

This engraver creates the following layout object(s): NoteName (page 597).

Note_name_engraver is part of the following context(s) in layout: NoteNames (page 226).
2.2.97 Note_performer

Music types accepted: articulation-event (page 49), breathing-event (page 50), note-event (page 54), and tie-event (page 58).

Note_performer is part of the following context(s) in \midi:
- ChordNames (page 95),
- CueVoice (page 97),
- GregorianTranscriptionVoice (page 153),
- KiwanVoice (page 189),
- MensuralVoice (page 216),
- PetrucciVoice (page 244),
- TabVoice (page 352),
- VaticanaVoice (page 380), and
- Voice (page 390).

2.2.98 Note_spacing_engraver

Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 598).

Note_spacing_engraver is part of the following context(s) in \layout:
- CueVoice (page 97),
- DrumVoice (page 117),
- GregorianTranscriptionVoice (page 153),
- KiwanVoice (page 189),
- MensuralVoice (page 216),
- PetrucciVoice (page 244),
- StandaloneRhythmVoice (page 331),
- TabVoice (page 352),
- VaticanaVoice (page 380), and
- Voice (page 390).

2.2.99 Ottava_spanner_engraver

Create a text spanner when the ottavation property changes.

Music types accepted: ottava-event (page 54),

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): OttavaBracket (page 599).

Ottava_spanner_engraver is part of the following context(s) in \layout:
- GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163),
- KiwanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), and
- VaticanaStaff (page 366).

2.2.100 Output_property_engraver

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 48),

Output_property_engraver is part of the following context(s) in \layout:
- ChoirStaff (page 66),
- ChordGrid (page 68), ChordGridScore (page 73),
- ChordNames (page 95),
- CueVoice (page 97),
- DrumStaff (page 108),
- DrumVoice (page 117),
- Dynamics (page 126),
- FretBoards (page 133),
- GrandStaff (page 135),
- GregorianTranscriptionStaff (page 140),
- GregorianTranscriptionVoice (page 153),
- InternalGregorianStaff (page 163),
- KiwanStaff (page 176),
- MensuralStaff (page 202),
- MensuralVoice (page 216),
- PetrucciStaff (page 231),
- PetrucciVoice (page 244),
- PianoStaff (page 255),
- RhythmicStaff (page 257),
- Score (page 263),
- Staff (page 287),
- StaffGroup (page 299),
- StandaloneRhythmScore (page 301),
- StandaloneRhythmStaff (page 325),
- StandaloneRhythmVoice (page 331),
- TabStaff (page 341),
- TabVoice (page 352),
- VaticanaStaff (page 366),
- VaticanaVoice (page 380), and
- Voice (page 390).
2.2.101 Page_turn_ engraver

Decide where page turns are allowed to go.

Music types accepted: break-event (page 49),

Properties (read)

- currentBarLine (graphical (layout) object)
  Set to the BarLine that Bar_ engraver has created in the current timestep.

- 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

2.2.102 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: break-event (page 49), and label-event (page 52),

Properties (read)

- forbidBreak (boolean)
  If set to #t, prevent a line break at this point, except if explicitly requested by the user.

Properties (write)

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

- 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, except if explicitly requested by the user.

- forceBreak (boolean)
  Set to #t when an event forcing a line break was heard.

This engraver creates the following layout object(s): NonMusicalPaperColumn (page 593), and PaperColumn (page 600).

Paper_column_ engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.103 Parenthesis_ engraver

Parenthesize objects whose parenthesize property is #t.

This engraver creates the following layout object(s): Parentheses (page 601).

Parenthesis_ engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).
2.2.104 Part_combine_engraver

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: note-event (page 54), and part-combine-event (page 54),

Properties (read)

\texttt{aDueText} (markup)
\hspace{1em} Text to print at a unisono passage.

\texttt{partCombineTextsOnNote} (boolean)
\hspace{1em} Print part-combine texts only on the next note rather than immediately on rests or skips.

\texttt{printPartCombineTexts} (boolean)
\hspace{1em} Set ‘Solo’ and ‘A due’ texts in the part combiner?

\texttt{soloIIText} (markup)
\hspace{1em} The text for the start of a solo for voice ‘two’ when part-combining.

\texttt{soloText} (markup)
\hspace{1em} The text for the start of a solo when part-combining.

This engraver creates the following layout object(s): CombineTextScript (page 517).

Part_combine_engraver is part of the following context(s) in \texttt{layout}: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.105 Percent_repeat_engraver

Make whole measure repeats.

Music types accepted: percent-event (page 54),

Properties (read)

\texttt{countPercentRepeats} (boolean)
\hspace{1em} If set, produce counters for percent repeats.

\texttt{currentCommandColumn} (graphical (layout) object)
\hspace{1em} Grob that is X-parent to all current breakable items (clef, key signature, etc.).

\texttt{repeatCountVisibility} (procedure)
\hspace{1em} A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when \texttt{countPercentRepeats} is set.

This engraver creates the following layout object(s): PercentRepeat (page 602), and PercentRepeatCounter (page 603).

Percent_repeat_engraver is part of the following context(s) in \texttt{layout}: ChordGrid (page 68), CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.106 Phrasing_slur_engraver

Print phrasing slurs. Similar to Section 2.2.126 [Slur_engraver], page 446.

Music types accepted: note-event (page 54), and phrasing-slur-event (page 55),

This engraver creates the following layout object(s): PhrasingSlur (page 604).
Phrasing_slur_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.107 Piano_pedal_align_engraver
Align piano pedal symbols and brackets.

Properties (read)

\begin{itemize}
\item currentCommandColumn (graphical (layout) object)
  \hfill Grob that is X-parent to all current breakable items (clef, key signature, etc.).
\end{itemize}

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 624), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

\texttt{Piano_pedal_align_engraver} is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.108 Piano_pedal_engraver
Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and una-corda-event (page 58),

Properties (read)

\begin{itemize}
\item currentCommandColumn (graphical (layout) object)
  \hfill Grob that is X-parent to all current breakable items (clef, key signature, etc.).
\item pedalSostenutoStrings (list)
  \hfill See pedalSustainStrings.
\item pedalSostenutoStyle (symbol)
  \hfill See pedalSustainStyle.
\item pedalSustainStrings (list)
  \hfill 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.
\item pedalSustainStyle (symbol)
  \hfill A symbol that indicates how to print sustain pedals: text, bracket or mixed (both).
\item pedalUnaCordaStrings (list)
  \hfill See pedalSustainStrings.
\item pedalUnaCordaStyle (symbol)
  \hfill See pedalSustainStyle.
\end{itemize}

This engraver creates the following layout object(s): PianoPedalBracket (page 606), SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

\texttt{Piano_pedal_engraver} is part of the following context(s) in \layout: Dynamics (page 126), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).
2.2.109 Piano_pedal_performer

Music types accepted: sostenuto-event (page 56), sustain-event (page 57), and
una-corda-event (page 58).

Piano_pedal_performer is part of the following context(s) in \midi: ChordNames (page 95), CueVoice (page 97), DrumVoice (page 117), Dynamics (page 126),
GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.110 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) in \layout: NullVoice (page 228), RhythmicStaff (page 257), and StandaloneRhythmStaff (page 325).

2.2.111 Pitched_trill_engraver

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), and TrillPitchParentheses (page 660).

Pitched_trill_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), VaticanaVoice (page 380), and Voice (page 390).

2.2.112 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) in \layout: DrumStaff (page 108), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), VaticanaLyrics (page 363), and VaticanaStaff (page 366).

2.2.113 Repeat_acknowledge_engraver

This translator adds entries to repeatCommands for events generated by \repeat volta.

Music types accepted: volta-repeat-end-event (page 59), and volta-repeat-start-event (page 59).

Properties (write)

  repeatCommands (list)
  A list of commands related to volta-style repeats. In general, each element is a list, 'command args...', but a command with no arguments may be abbreviated to a symbol; e.g., '(start-repeat) may be given as '(start-repeat).

end-repeat
  End a repeated section.
start-repeat
    Start a repeated section.

volta text
    If text is markup, start a volta bracket with that label; if text is #f, end a
    volta bracket.

Repeat_acknowledge_engraver is part of the following context(s) in \layout:
ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.114 Repeat_tie_engraver
Create repeat ties.

Music types accepted: repeat-tie-event (page 55),

This engraver creates the following layout object(s): RepeatTie (page 610), and
RepeatTieColumn (page 611).

Repeat_tie_engraver is part of the following context(s) in \layout: CueVoice (page 97),
DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189),
MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331),
TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.115 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): RestCollision (page 612).

Rest_collision_engraver is part of the following context(s) in \layout: DrumStaff
(page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff
(page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231),
Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.116 Rest_engraver
Engrave rests.

Music types accepted: rest-event (page 55),

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): Rest (page 611).

Rest_engraver is part of the following context(s) in \layout: CueVoice (page 97),
DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189),
MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331),
TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).
2.2.117 **Rhythmic_column_engraver**

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): `NoteColumn` (page 595).

*Rhythmic_column_engraver* is part of the following context(s) in `\layout`: `CueVoice` (page 97), `DrumVoice` (page 117), `GregorianTranscriptionVoice` (page 153), `KievanVoice` (page 189), `MensuralVoice` (page 216), `PetrucciVoice` (page 244), `StandaloneRhythmVoice` (page 331), `TabVoice` (page 352), `VaticanaVoice` (page 380), and `Voice` (page 390).

2.2.118 **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): `ScriptColumn` (page 614).

*Script_column_engraver* is part of the following context(s) in `\layout`: `CueVoice` (page 97), `DrumVoice` (page 117), `GregorianTranscriptionVoice` (page 153), `KievanVoice` (page 189), `MensuralVoice` (page 216), `PetrucciVoice` (page 244), `StandaloneRhythmVoice` (page 331), `TabVoice` (page 352), `VaticanaVoice` (page 380), and `Voice` (page 390).

2.2.119 **Script_engraver**

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 49),

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): `Script` (page 613).

*Script_engraver* is part of the following context(s) in `\layout`: `CueVoice` (page 97), `DrumVoice` (page 117), `Dynamics` (page 126), `GregorianTranscriptionVoice` (page 153), `KievanVoice` (page 189), `MensuralVoice` (page 216), `PetrucciVoice` (page 244), `StandaloneRhythmVoice` (page 331), `TabVoice` (page 352), `VaticanaVoice` (page 380), and `Voice` (page 390).

2.2.120 **Script_row_engraver**

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): `ScriptRow` (page 614).

*Script_row_engraver* is part of the following context(s) in `\layout`: `DrumStaff` (page 108), `GregorianTranscriptionStaff` (page 140), `InternalGregorianStaff` (page 163), `KievanStaff` (page 176), `MensuralStaff` (page 202), `PetrucciStaff` (page 231), `Staff` (page 287), `TabStaff` (page 341), and `VaticanaStaff` (page 366).

2.2.121 **Separating_line_group_engraver**

Generate objects for computing line spacing parameters.

Properties (read)

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

Properties (write)

- `hasStaffSpacing` (boolean)
  - True if `currentCommandColumn` contains items that will affect spacing.
This engraver creates the following layout object(s): StaffSpacing (page 632).

Separating_line_group_engraver is part of the following context(s) in \layout: ChordNames (page 95), DrumStaff (page 108), FiguredBass (page 132), FretBoards (page 133), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), NoteNames (page 226), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.122 Show_control_points_engraver

Create grobs to visualize control points of Bézier curves (ties and slurs) for ease of tweaking.

This engraver creates the following layout object(s): ControlPoint (page 519), and ControlPolygon (page 520).

Show_control_points_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.123 Signum_repetitionis_engraver

Create a SignumRepetitionis at the end of a \repeat volta section.

Music types accepted: volta-repeat-end-event (page 59),

This engraver creates the following layout object(s): SignumRepetitionis (page 618).

Signum_repetitionis_engraver is part of the following context(s) in \layout: PetrucciStaff (page 231).

2.2.124 Skip_typesetting_engraver

Create a StaffEllipsis when skipTypesetting is used.

Properties (read)

skipTypesetting (boolean)

If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

This engraver creates the following layout object(s): StaffEllipsis (page 628).

Skip_typesetting_engraver is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.125 Slash_repeat_engraver

Make beat repeats.

Music types accepted: repeat-slash-event (page 55),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

Slash_repeat_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.126 Slur_engraver

Build slur grobs from slur events.

Music types accepted: note-event (page 54), and slur-event (page 56),
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): Slur (page 621).

Slur_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), NullVoice (page 228), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), and Voice (page 390).

2.2.127 Slur_performer
Music types accepted: slur-event (page 56),

Slur_performer is part of the following context(s) in \midi: ChordNames (page 95), CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), NullVoice (page 228), PetrucciVoice (page 244), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.128 Spacing_engraver
Make a SpacingSpanner and do bookkeeping of shortest starting and playing notes.

Music types accepted: spacing-section-event (page 56),

Properties (read)

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

  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): SpacingSpanner (page 626).

Spacing_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.129 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): Arpeggio (page 483).

Span_arpeggio_engraver is part of the following context(s) in \layout: ChoirStaff (page 66), GrandStaff (page 135), PianoStaff (page 255), and StaffGroup (page 299).
2.2.130 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): SpanBar (page 626).

Span_bar_engraver is part of the following context(s) in \layout: GrandStaff (page 135), PianoStaff (page 255), and StaffGroup (page 299).

2.2.131 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): SpanBarStub (page 628).

Span_bar_stub_engraver is part of the following context(s) in \layout: ChoirStaff (page 66), GrandStaff (page 135), PianoStaff (page 255), and StaffGroup (page 299).

2.2.132 Span.stem_engraver
Connect cross-staff stems to the stems above in the system

This engraver creates the following layout object(s): Stem (page 634).

Span.stem_engraver is not part of any context

2.2.133 Spanner_break_forbid_engraver
Forbid breaks in certain spanners.

Spanner_break_forbid_engraver is part of the following context(s) in \layout:
CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153),
KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244),
StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and
Voice (page 390).

2.2.134 Spanner_tracking_engraver
Helper for creating spanners attached to other spanners. If a spanner has the sticky-grob-interface, the engraver tracks the spanner contained in its sticky-host object. When the host ends, the sticky spanner attached to it has its end announced too.

Spanner_tracking_engraver is part of the following context(s) in \layout:
ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.135 Staff_collecting_engraver
Maintain the stavesFound variable.

Properties (read)

\begin{itemize}
  \item stavesFound (list of grobs)
    \begin{itemize}
      \item A list of all staff-symbols found.
    \end{itemize}
\end{itemize}

Properties (write)

\begin{itemize}
  \item stavesFound (list of grobs)
    \begin{itemize}
      \item A list of all staff-symbols found.
    \end{itemize}
\end{itemize}

Staff_collecting_engraver is part of the following context(s) in \layout:
ChordGridScore (page 73), DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), Score (page 263), Staff (page 287), StandaloneRhythmScore (page 301), TabStaff (page 341), and VaticanaStaff (page 366).
2.2.136 Staff_highlight_engraver
Highlights music passages.

Music types accepted: staff-highlight-event (page 56),
Properties (read)

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

This engraver creates the following layout object(s): StaffHighlight (page 631).

Staff_highlight_engraver is part of the following context(s) in \layout: DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.137 Staff_performer
Properties (read)

  midiChannelMapping (symbol)
   How to map MIDI channels: per staff (default), instrument or voice.

  midiMergeUnisons (boolean)
   If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

  midiSkipOffset (moment)
   This is the accrued MIDI offset to account for time skipped via skipTypesetting.

Staff_performer is part of the following context(s) in \midi: ChordNames (page 95), DrumStaff (page 108), GregorianTranscriptionLyrics (page 137), GregorianTranscriptionStaff (page 140), KievanStaff (page 176), Lyrics (page 199), MensuralStaff (page 202), NoteNames (page 226), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.138 Staff_symbol_engraver
Create the constellation of five (default) staff lines.

Music types accepted: staff-span-event (page 57),

This engraver creates the following layout object(s): StaffSymbol (page 632).

Staff_symbol_engraver is part of the following context(s) in \layout: ChordGrid (page 68), DrumStaff (page 108), GregorianTranscriptionStaff (page 140), InternalGregorianStaff (page 163), KievanStaff (page 176), MensuralStaff (page 202), PetrucciStaff (page 231), RhythmicStaff (page 257), Staff (page 287), StandaloneRhythmStaff (page 325), TabStaff (page 341), and VaticanaStaff (page 366).

2.2.139 Stanza_number_align_engraver
This engraver ensures that stanza numbers are neatly aligned.

Stanza_number_align_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).
2.2.140 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): StanzaNumber (page 633).

Stanza_number_engraver is part of the following context(s) in \layout:
GregorianTranscriptionLyrics (page 137), Lyrics (page 199), and VaticanaLyrics (page 363).

2.2.141 Stem_engraver

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: tremolo-event (page 58), and tuplet-span-event (page 58),

Properties (read)

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

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.

This engraver creates the following layout object(s): Flag (page 547), Stem (page 634), StemStub (page 636), and StemTremolo (page 637).

Stem_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), and Voice (page 390).

2.2.142 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 items (clef, key signature, etc.).

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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

System_start_delimiter_engraver is part of the following context(s) in \layout: ChoirStaff (page 66), ChordGrid (page 68), GrandStaff (page 135), PianoStaff (page 255), Score (page 263), StaffGroup (page 299), and StandaloneRhythmScore (page 301).
### 2.2.143 Tab_note_heads_engraver

Generate one or more tablature note heads from event of type `NoteEvent`.

Music types accepted: fingering-event (page 51), note-event (page 54), and string-number-event (page 57).

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.
- **maximumFretStretch** (number)
  
  Don’t allocate frets further than this from specified frets.
- **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): TabNoteHead (page 648).

Tab_note_heads_engraver is part of the following context(s) in `layout`: TabVoice (page 352).

### 2.2.144 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): StaffSymbol (page 632).

Tab_staff_symbol_engraver is part of the following context(s) in \layout: TabStaff (page 341).

2.2.145 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) in \layout: TabVoice (page 352).

2.2.146 Tempo_performer
Properties (read)

   tempoWholesPerMinute (moment)
   The tempo in whole notes per minute.

Tempo_performer is part of the following context(s) in \midi: Score (page 263).

2.2.147 Text_engraver
Create text scripts.

Music types accepted: text-script-event (page 58),

This engraver creates the following layout object(s): TextScript (page 650).

Text_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), Dynamics (page 126), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.148 Text_spanner_engraver
Create text spanner from an event.

Music types accepted: text-span-event (page 58),

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): TextSpanner (page 652).

Text_spanner_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), Dynamics (page 126), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), and Voice (page 390).

2.2.149 Tie_engraver
Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 58),

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)

\texttt{tieMelismaBusy} (boolean)

Signal whether a tie is present.

This engraver creates the following layout object(s): \texttt{Tie} (page 653), and \texttt{TieColumn} (page 655).

\texttt{Tie\_engraver} is part of the following context(s) in \texttt{layout}: \texttt{CueVoice} (page 97), \texttt{DrumVoice} (page 117), \texttt{GregorianTranscriptionVoice} (page 153), \texttt{KievanVoice} (page 189), \texttt{MensuralVoice} (page 216), \texttt{NoteNames} (page 226), \texttt{NullVoice} (page 228), \texttt{PetrucciVoice} (page 244), \texttt{StandaloneRhythmVoice} (page 331), \texttt{TabVoice} (page 352), \texttt{VaticanaVoice} (page 380), and \texttt{Voice} (page 390).

\subsection*{2.2.150 Tie\_performer}

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 58),

Properties (read)

\texttt{tieWaitForNote} (boolean)

If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

\texttt{tieMelismaBusy} (boolean)

Signal whether a tie is present.

\texttt{Tie\_performer} is part of the following context(s) in \texttt{midi}: \texttt{ChordNames} (page 95), \texttt{CueVoice} (page 97), \texttt{DrumVoice} (page 117), \texttt{GregorianTranscriptionVoice} (page 153), \texttt{KievanVoice} (page 189), \texttt{MensuralVoice} (page 216), \texttt{NullVoice} (page 228), \texttt{PetrucciVoice} (page 244), \texttt{TabVoice} (page 352), \texttt{VaticanaVoice} (page 380), and \texttt{Voice} (page 390).

\subsection*{2.2.151 Time\_signature\_engraver}

Create a Section 3.1.146 \texttt{[TimeSignature]}, page 655, whenever \texttt{timeSignatureFraction} changes.

Music types accepted: time-signature-event (page 58),

Properties (read)

\texttt{initialTimeSignatureVisibility} (vector)

break visibility for the initial time signature.

\texttt{partialBusy} (boolean)

Signal that \texttt{partial} acts at the current timestep.

\texttt{timeSignatureFraction} (fraction, as pair)

A pair of numbers, signifying the time signature. For example, \texttt{'(4 \ . \ 4)} is a 4/4 time signature.

This engraver creates the following layout object(s): \texttt{TimeSignature} (page 655).

\texttt{Time\_signature\_engraver} is part of the following context(s) in \texttt{layout}: \texttt{DrumStaff} (page 108), \texttt{InternalGregorianStaff} (page 163), \texttt{MensuralStaff} (page 202), \texttt{PetrucciStaff} (page 231), \texttt{RhythmicStaff} (page 257), \texttt{Staff} (page 287), and \texttt{TabStaff} (page 341).
2.2.152 **Time_signature_performer**

Creates a MIDI time signature whenever `timeSignatureFraction` changes or a \time command is issued.

Music types accepted: time-signature-event (page 58),

Properties (read)

`timeSignatureFraction` (fraction, as pair)
A pair of numbers, signifying the time signature. For example, '(4 . 4) is a 4/4 time signature.

`Time_signature_performer` is part of the following context(s) in \midi: Score (page 263).

2.2.153 **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.

Music types accepted: alternative-event (page 48), bar-event (page 49), and fine-event (page 51),

Properties (read)

`alternativeNumberingStyle` (symbol)
The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

`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_engineer`.

`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)

`alternativeNumber` (non-negative, exact integer)
When set, the index of the current \alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple volte.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.
currentBarNumber (integer)
Contains the current bar number. This property is incremented at every bar line.

internalBarNumber (integer)
Contains the current bar number. 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.

measureStartNow (boolean)
True at the beginning of a measure.

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) in \layout: ChordGridScore
(page 73), Score (page 263), and StandaloneRhythmScore (page 301); in \midi: Score
(page 263).

2.2.154 Trill_spanner_engraver
Create trill spanners.

Music types accepted: trill-span-event (page 58),

Properties (read)

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

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): TrillSpanner (page 661).

Trill_spanner_engraver is part of the following context(s) in \layout: CueVoice
(page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice
(page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice
(page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.155 Tuplet_engraver
Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 58),

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): TupletBracket (page 662), and
TupletNumber (page 664).
Tuplet_engraver is part of the following context(s) in \layout: CueVoice (page 97), DrumVoice (page 117), GregorianTranscriptionVoice (page 153), KievanVoice (page 189), MensuralVoice (page 216), PetrucciVoice (page 244), StandaloneRhythmVoice (page 331), TabVoice (page 352), VaticanaVoice (page 380), and Voice (page 390).

2.2.156 Tweak_engraver
Read the tweaks property from the originating event, and set properties.

Tweak_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

2.2.157 Vaticana_ligature_engraver
Handle ligatures by gluing special ligature heads together.

Music types accepted: ligature-event (page 52), and pes-or-flexa-event (page 55),

This engraver creates the following layout object(s): DotColumn (page 531), and VaticanaLigature (page 667).

Vaticana_ligature_engraver is part of the following context(s) in \layout: VaticanaVoice (page 380).

2.2.158 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): StaffGrouper (page 630), and VerticalAlignment (page 668).

Vertical_align_engraver is part of the following context(s) in \layout: ChoirStaff (page 66), ChordGridScore (page 73), GrandStaff (page 135), PianoStaff (page 255), Score (page 263), StaffGroup (page 299), and StandaloneRhythmScore (page 301).

2.2.159 Volta_engraver
Make volta brackets.

Music types accepted: dal-segno-event (page 50), fine-event (page 51), and volta-span-event (page 59),

Properties (read)

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

repeatCommands (list)
A list of commands related to volta-style repeats. In general, each element is a list, '(command args ...)’, but a command with no arguments may be abbreviated to a symbol; e.g., ‘(start-repeat)’ may be given as ‘(start-repeat).

end-repeat
End a repeated section.
start-repeat
Start a repeated section.

volta text
If text is markup, start a volta bracket with that label; if text is \#f, end a volta bracket.

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): VoltaBracket (page 671), and VoltaBracketSpanner (page 672).

Volta_engraver is part of the following context(s) in \layout: ChordGridScore (page 73), Score (page 263), and StandaloneRhythmScore (page 301).

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 separate 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.

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

alterationGlyphs (list)
A list mapping alterations to accidental glyphs. Alterations are given as exact numbers, e.g., -1/2 for flat. This applies to all grobs that can print accidentals.

alternativeNumber (non-negative, exact integer)
When set, the index of the current \alternative element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple voltes.

alternativeNumberingStyle (symbol)
The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to numbers to reset the bar number at each alternative, or set to numbers-with-letters to reset and also include letter suffixes.

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.
   
      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.

barCheckSynchronize (boolean)
   If true then reset measurePosition when finding a bar check.

barExtraVelocity (integer)
   Extra MIDI velocity added by the ‘Beat_performer’ at the start of each measure.

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.

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.

beatExtraVelocity (integer)
Extra MIDI velocity added by the ‘Beat_performer’ at the start of each beat.

beatStructure (list)
List of baseMoments that are combined to make beats.

breathMarkType (symbol)
The type of BreathingSign to create at \breathe.

caesuraType (list)
An alist

((bar-line . bar-type)
(breath . breath-type)
(script . script-type...)
(underlying-bar-line . bar-type))
specifying which breath mark, bar line, and scripts to create at \caesura. All entries are optional.

bar-line has higher priority than a measure bar line and underlying-bar-line has lower priority than a measure bar line.
caesuraTypeTransform (procedure)
An engraver callback taking three arguments and returning an alist of the same kind as caesuraType.
The first argument is the context.
The second argument is the value of caesuraType with an additional entry (articulations . symbol-list) identifying the articulations attached to the caesura in the music. If the transform function returns this second argument unmodified, it is as if no transform function were set; the function is free to return a different value. The transform function can remove articulations, but any added articulations are ignored.
The third argument is a symbol-list identifying certain things the engraver has observed. bar-line indicates that the engraver has observed a BarLine at the current moment.

centerBarNumbers (boolean)
Whether to center bar numbers in their measure instead of aligning them on the bar line.

chordChanges (boolean)
Only show changes in chords scheme?

chordNameExceptions (list)
An alist of chord exceptions. Contains (chord . 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.

chordNamer (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)
The 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’.

codaMarkFormatter (procedure)
A procedure that creates a coda mark (which in conventional D.S. at Coda form indicates the start of the alternative endings), taking as arguments the mark sequence number and the context. It should return a markup object.
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 bar number. This property is incremented at every bar line.
dalSegnoTextFormatter (procedure)
Format a jump instruction such as D.S.
The first argument is the context.
The second argument is the number of times the instruction is performed.
The third argument is a list of three markups: start-markup, end-markup, and next-markup.
If \textit{start-markup} is \texttt{#f}, the form is \textit{da capo}; otherwise the form is \textit{dal segno} and \textit{start-markup} is the sign at the start of the repeated section.

If \textit{end-markup} is not \texttt{#f}, it is either the sign at the end of the main body of the repeat, or it is a \textit{Fine} instruction. When it is a Fine instruction, \textit{next-markup} is \texttt{#f}.

If \textit{next-markup} is not \texttt{#f}, it is the mark to be jumped to after performing the body of the repeat, e.g., Coda.

decrescendoSpanner (symbol)
The type of spanner to be used for decrescendi. Available values are ‘\texttt{hairpin}’ and ‘\texttt{text}’. If unset, a hairpin decrescendo is used.

decrescendoText (markup)
The text to print at start of non-hairpin decrescendo, i.e., ‘\texttt{dim.}’.

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.

doubleRepeatBarType (string)
Bar line to insert where the end of one \texttt{\repeat volta} coincides with the start of another. The default is ‘\texttt{:..:}’.

doubleRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of one \texttt{\repeat volta} and the beginning of another. The default is ‘\texttt{:|\|.S.|:}’.

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)

The layout style is a hash table, containing the drum-pitches (e.g., the symbol ‘hihat’) as keys, and a list (\texttt{notehead-style script vertical-position}) as values.

eadAtSkip (boolean)
End DurationLine grob on skip-event

eadRepeatBarType (string)
Bar line to insert at the end of a \texttt{\repeat volta}. The default is ‘\texttt{:|\|.}’.

eadRepeatSegnoBarType (string)
Bar line to insert where an in-staff segno coincides with the end of a \texttt{\repeat volta}. The default is ‘\texttt{:|\|.S.|:}’.

\texttt{explicitClefVisibility} (vector)
‘break-visibility’ function for clef changes.

\texttt{explicitCueClefVisibility} (vector)
‘break-visibility’ function for cue clef changes.

\texttt{explicitKeySignatureVisibility} (vector)
‘break-visibility’ function for explicit key changes. ‘\texttt{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.
figuredBassLargeNumberAlignment (number)
   Horizontal alignment to use for numbers in figured bass that contain more than a single
digit.
figuredBassPlusDirection (direction)
   Where to put plus signs relative to the main figure.
figuredBassPlusStrokedAlist (list)
   An alist mapping figured bass digits to glyphs. The default is mapping numbers 2, 4, 5, 6,
   7, and 9 to the six glyphs figbass.*plus and figbass.*stroked, respectively.
finalFineTextVisibility (boolean)
   Whether \fine at the written end of the music should create a Fine instruction.
fineBarType (string)
   Bar line to insert at \fine. Where there is also a repeat bar line, the repeat bar line takes
   precedence and this value is appended to it as an annotation. The default is '|.'.
fineSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with \fine. The default is '|.S'.
fineStartRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with \fine and the start of a \repeat
   volta. The default is '|.S.|:'.
fineText (markup)
   The text to print at \fine.
fingeringsOrientations (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, except if explicitly requested by the user.
forbidBreakBetweenBarLines (boolean)
   If set to true, Bar_engraver forbids line breaks where there is no bar line.
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 GridPoints.

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.

ignoreBarNumberChecks (boolean)
  Ignore bar number 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.
  This property is deprecated

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 quotes.

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)
A list of pairs that defines in what order alterations should be printed. The format of an entry is (step . alter), where step is a number from 0 to 6 and alter from -1 (double flat) to 1 (double sharp), with exact rationals for alterations in between, e.g., 1/2 for sharp.

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.

lyricRepeatCountFormatter (procedure)
A procedure taking as arguments the context and the numeric repeat count. It should return the formatted repeat count as markup. If it does not return markup, no grob is created.

magnifyStaffValue (positive number)
The most recent value set with \magnifyStaff.

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

maximumFretStretch (number)
Don’t allocate frets further than this from specified frets.

measureBarType (string)
Bar line to insert at a measure boundary.

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.

measureStartNow (boolean)
True at the beginning of a measure.

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.

noteNameFunction (procedure)
Function used to convert pitches into strings and markups.

noteNameSeparator (string)
String used to separate simultaneous NoteName objects.

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.

ottavaStartNow (boolean)
Is an ottava starting in this time step?

ottavation (markup)
If set, the text for an ottava spanner. Changing this creates a new text spanner.

ottavationMarkups (list)
An alist defining the markups used for ottava brackets. It contains entries of the form (number of octaves . markup).

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.

printAccidentalNames (boolean or symbol)
    Print accidentals in the NoteNames context.

printKeyCancellation (boolean)
    Print restoration alterations before a key signature change.

printNotesLanguage (string)
    Use a specific language in the NoteNames context.

printOctaveNames (boolean or symbol)
    Print octave marks in 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 next rehearsal mark to print.

rehearsalMarkFormatter (procedure)
    A procedure taking as arguments the context and the sequence number of the rehearsal mark. It should return the formatted mark as a markup object.

duplicateCommands (list)
    A list of commands related to volta-style repeats. In general, each element is a list, ' (command args...), but a command with no arguments may be abbreviated to a symbol; e.g., '((start-repeat)) may be given as '(start-repeat).

end-repeat
    End a repeated section.

start-repeat
    Start a repeated section.

volta text
    If text is markup, start a volta bracket with that label; if text is #f, end a volta bracket.

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.
sectionBarType (string)
  Bar line to insert at \section. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is ‘||’.

segnoBarType (string)
  Bar line to insert at an in-staff segno. The default is ‘S’.

segnoMarkFormatter (procedure)
  A procedure that creates a segno (which conventionally indicates the start of a repeated section), taking as arguments the mark sequence number and the context. It should return a markup object.

segnoStyle (symbol)
  A symbol that indicates how to print a segno: bar-line or mark.

shapeNoteStyles (vector)
  Vector of symbols, listing style for each note head relative to the tonic (q.v.) 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.

startAtNoteColumn (boolean)
  Start DurationLine grob at entire NoteColumn.
startAtSkip (boolean)
   Start DurationLine grob at skip-event.

startRepeatBarType (string)
   Bar line to insert at the start of a \repeat volta. The default is ‘.|.’.

startRepeatSegnoBarType (string)
   Bar line to insert where an in-staff segno coincides with the start of a \repeat volta. The default is ‘S.|.’.

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 or symbol)
   If set to #t, accidentals are typeset as suggestions above the note. Setting it to 'cautionary only applies that to cautionary accidentals.

supportNonIntegerFret (boolean)
   If set in Score the TabStaff will print micro-tones as ‘2\frac{1}{2}’

suspendMelodyDecisions (boolean)
   When using the Melody_engraver, stop changing orientation of stems based on the melody when this is set to true.

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 \} 
\}

underlyingRepeatBarType (string)
Bar line to insert at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno. Where there is also a repeat bar line, the repeat bar line takes precedence and this value is appended to it as an annotation. The default is '||'.

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)
The current bar line type, or '()' if there is no bar line. Setting this explicitly in user code is deprecated. Use \bar or related commands to set it.

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.

breathMarkDefinitions (list)
The description of breath marks. This is used by the Breathing_sign_engraver. See scm/breath.scm for more information.

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.).

codaMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a coda mark appears: not set during the first timestep, 0 up to the first coda mark, 1 from the first to the second, 2 from the second to the third, etc.

currentBarLine (graphical (layout) object)
Set to the BarLine that Bar_engraver has created in the current timestep.

currentChordCause (stream event)
Event cause of the chord that should be created in this time step (if any).

currentChordText (markup)
In contexts printing chord names, this is at any point of time the markup that will be put in the chord name.

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

currentMusicalColumn (graphical (layout) object)
Grob that is X-parent to all non-breakable items (note heads, lyrics, etc.).

currentPerformanceMarkEvent (stream event)
The coda, section, or segno mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

currentRehearsalMarkEvent (stream event)
The ad-hoc or rehearsal mark event selected by Mark_tracking_translator for engraving by Mark_engraver.

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.

forceBreak (boolean)
Set to #t when an event forcing a line break was heard.

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 currentCommandColumn 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.

midiSkipOffset (moment)
This is the accrued MIDI offset to account for time skipped via skipTypesetting.

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.

segnoMarkCount (non-negative, exact integer)
Updated at the end of each timestep in which a segno appears: not set during the first timestep, 0 up to the first segno, 1 from the first to the second segno, 2 from the second to the third segno, etc.

slurMelismaBusy (boolean)
Signal if a slur is present.
stavesFound (list of grobs)
   A list of all staff-symbols found.

stringFretFingerList (list)
   A list containing three entries. In TabVoice and FretBoards they determine the string, fret and finger to use

tieMelismaBusy (boolean)
   Signal whether a tie is present.
3 Backend

3.1 All layout objects

3.1.1 Accidental

An accidental. Horizontal padding and configuration between accidentals is controlled by the AccidentalPlacement (page 477), grob.

Accidental objects are created by: Accidental_engraver (page 400).

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).

horizontal-skylines (pair of skylines):
    #<unpure-pure-container #<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 #<procedure ly:grob::vertical-skylines-from-stencil (_)>> #<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):

```lisp
#<unpure-pure-container #<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): accidental-interface (page 675), accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), inline-accidental-interface (page 710), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.2 AccidentalCautionary

A cautionary accidental, normally enclosed in parentheses.

AccidentalCautionary objects are created by: Accidental_engraver (page 400).

Standard settings:

after-line-breaking (boolean):

```lisp
ly:accidental-interface::remove-tied
```

Dummy property, used to trigger callback for after-line-breaking.

alteration (number):

```lisp
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).

horizontal-skylines (pair of skylines):

```lisp
#<unpure-pure-container #<procedure ly:accidental-interface::horizontal-skylines (_)> >
```

Two skylines, one to the left and one to the right of this grob.

parenthesized (boolean):

`#t`

Parenthesize this grob.

stencil (stencil):

```lisp
ly:accidental-interface::print
```

The symbol to print.
vertical-skylines (pair of skylines):

```
#<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<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 #<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): accidental-interface (page 675), accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), inline-accidental-interface (page 710), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.3 AccidentalPlacement

In groups of Accidental (page 475), grobs, this auxiliary grob controls their horizontal padding and configuration (which ones are placed more to left or to the right).

AccidentalPlacement objects are created by: Accidental_engraver (page 400), and Ambitus_engraver (page 402).

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): accidental-placement-interface (page 675), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).
3.1.4 AccidentalSuggestion

An annotational accidental as used in *musica ficta*. Normally positioned above a note.

AccidentalSuggestion objects are created by: Accidental_engraver (page 400).

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.

- **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 <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 <procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >```

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-interface (page 675), accidental-suggestion-interface (page 676), accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), script-interface (page 734), self-alignment-interface (page 736), and side-position-interface (page 738).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.5 Ambitus

An ambitus, giving the range of pitches of a voice or instrument. It aligns AmbitusAccidental (page 481), AmbitusLine (page 481), and AmbitusNoteHead (page 482), horizontally and defines the horizontal spacing from the ambitus to other items.

Ambitus objects are created by: Ambitus_engraver (page 402).

**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, `#:f` unbroken `begin-of-line`  `#:t` means visible, `#:f` means killed.
non-musical (boolean):
  #t
  True if the grob belongs to a NonMusicalPaperColumn.

space-alist (alist, with symbols as keys):
  '((cue-end-clef extra-space . 0.5)
   (clef extra-space . 1.15)
   (cue-clef extra-space . 0.5)
   (key-signature extra-space . 1.15)
   (signum-repetitionis extra-space . 1.15)
   (staff-bar extra-space . 1.15)
   (time-signature extra-space . 1.15)
   (right-edge extra-space . 0.5)
   (first-note extra-space . 1.15))

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.
Chapter 3: Backend

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):
ly:axis-group-interface::height
(#<unpure-pure-container #<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): ambitus-interface (page 677), axis-group-interface (page 678), break-aligned-interface (page 687), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.6 AmbitusAccidental

An accidental in an Ambitus (page 479).

AmbitusAccidental objects are created by: Ambitus_engraver (page 402).

Standard settings:

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):
ly:accidental-interface::height
(#<unpure-pure-container #<procedure ly:accidental-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): accidental-interface (page 675), accidental-switch-interface (page 676), break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.7 AmbitusLine

The vertical line in an Ambitus (page 479).

AmbitusLine objects are created by: Ambitus_engraver (page 402).

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): ambitus-interface (page 677), font-interface (page 698), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.8 AmbitusNoteHead
A note head in an Ambitus (page 479).

AmbitusNoteHead objects are created by: Ambitus_engraver (page 402).

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):

\[
\text{Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.}
\]

Y-offset (number):

\[
The vertical amount that this object is moved relative to its Y-parent.
\]

This object supports the following interface(s): ambitus-interface (page 677), font-interface (page 698), grob-interface (page 704), item-interface (page 712), ledgered-interface (page 715), note-head-interface (page 727), rhythmic-head-interface (page 734), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.9 Arpeggio

An arpeggio line (normally a vertical wiggle).

Arpeggio objects are created by: Arpeggio_engraver (page 402), and Span_arpeggio_engraver (page 447).

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 (start . end), where start and end are vertical positions 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 \textit{X} (or equivalently 0), the object is placed horizontally next to the other object. If the value is \textit{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 \texttt{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 #<procedure ly:grob::stencil-height (_)> #<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 #<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): arpeggio-interface (page 677), font-interface (page 698), grob-interface (page 704), item-interface (page 712), side-position-interface (page 738), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.10 BalloonText

A balloon text with a pointing line to visually mark and annotate another grob.

BalloonText objects are created by: Balloon_engraver (page 404).

Standard settings:

after-line-breaking (boolean):
  ly:balloon-interface::remove-irrelevant-spanner
  Dummy property, used to trigger callback for after-line-breaking.
Chapter 3: Backend

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.

break-visibility (vector):
  #<procedure 5575a26ab980 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:2973:0
  (grob)>
  A vector of 3 booleans, #(end-of-line unbroken begin-of-line). #t means visible,
  #f means killed.

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 5575a26ab940 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0
  (grob)>
  Text markup. See Section “Formatting text” in Notation Reference.

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 #<procedure ly:grob::vertical-skylines-from-
  stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-
  extents (_ _ _)>>
  Two skylines, one above and one below this grob.

X-extent (pair of numbers):
  ly:balloon-interface::width
  Extent (size) in the X direction, measured in staff-space units, relative to object’s
  reference point.

X-offset (number):
  #<procedure 5575a26ab920 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:
  (grob)>
  The horizontal amount that this object is moved relative to its X-parent.
Y-extent (pair of numbers):
   #<unpure-pure-container #<procedure ly:grob::stencil-height (_)> #<procedure ly:balloon-interface::pure-height (_ _ _)>>
   Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

Y-offset (number):
   #<procedure 5575a26ab900 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface (page 676), balloon-interface (page 680), font-interface (page 698), grob-interface (page 704), sticky-grob-interface (page 752), and text-interface (page 755).

This object can be of either of the following classes: Item (characterized by item-interface) or Spanner (characterized by spanner-interface). It supports the following interfaces conditionally depending on the class: item-interface (page 712), and spanner-interface (page 746).

3.1.11 BarLine

A bar line.

BarLine objects are created by: Bar_engraver (page 404).

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 \((-\infty, 0 \ . \ +\infty)\).

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-left (string):
#f
The glyph value to use at the end of the line when the line is broken. #f indicates that no glyph should be visible; otherwise the value must be a string.

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.

glyph-right (string):
#f
The glyph value to use at the beginning of the line when the line is broken. #f indicates that no glyph should be visible; otherwise the value must be a string.

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).

horizontal-skylines (pair of skylines):
#<unpure-pure-container #<procedure ly:grob::horizontal-skylines-from-stencil (_)> >
Two skylines, one to the left and one to the right of this grob.

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):
#f
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 (alist, with symbols as keys):
'((ambitus extra-space . 1.0)
 (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 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: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 #<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): bar-line-interface (page 681), break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), and pure-from-neighbor-interface (page 733).

This object is of class Item (characterized by item-interface (page 712)).

3.1.12 BarNumber
An ordinary bar number. Centered bar numbers are managed separately with CenteredBarNumber (page 506), grobs.

BarNumber objects are created by: Bar_number_engraver (page 406).

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):
  #<procedure 5575a25ef150 at /build/out/share/lilypond/current/scm/lily/output-lib.scm (grob)>
  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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): bar-number-interface (page 682), break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.13 BassFigure
A number in figured bass. It can contain an alteration as well.

BassFigure objects are created by: Figured_bass_engraver (page 421).

Standard settings:

  font-features (list):
  '("tnum" "cv47" "ss01")
  Opentype features.

stencil (stencil):
  ly:text-interface::print
  The symbol to print.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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): accidental-switch-interface (page 676), bass-figure-interface (page 682), font-interface (page 698), grob-interface (page 704), item-interface (page 712), rhythmic-grob-interface (page 734), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).
3.1.14 BassFigureAlignment

An auxiliary grob to stack several BassFigureLine (page 494), grobs vertically.

BassFigureAlignment objects are created by: Figured_bass_engraver (page 421).

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):
  -inf.0
  Add this much extra space between objects that are next to each other.

stacking-dir (direction):
  -1
  Stack objects in which direction?

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 #<procedure ly:axis-group-interface::height (_)> #<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): align-interface (page 676),
axis-group-interface (page 678), bass-figure-alignment-interface (page 682),
grob-interface (page 704), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.15 BassFigureAlignmentPositioning

If figured bass is used in the Staff (page 287), context, this auxiliary grob groups all of the
figured bass notation and computes an offset from the staff via side-positioning.

BassFigureAlignmentPositioning objects are created by: Figured_bass_position_engraver (page 422).

Standard settings:

add-stem-support (boolean):
  #t
  If set, the Stem object is included in this script’s support.

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 #<procedure ly:axis-group-interface::height (_)> #<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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): axis-group-interface (page 678), grob-interface (page 704), outside-staff-interface (page 730), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.16 BassFigureBracket

Brackets around a figured bass (or elements of it).

BassFigureBracket objects are created by: Figured_bass_engraver (page 421).

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): enclosing-bracket-interface (page 695), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.17 BassFigureContinuation
A horizontal line to indicate that a number of a previous figured bass is continued in the current figured bass.

BassFigureContinuation objects are created by: Figured_bass_engraver (page 421).

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): figured-bass-continuation-interface (page 696), grob-interface (page 704), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.18 BassFigureLine
An auxiliary grob providing a baseline for bass figures that should be aligned vertically.

BassFigureLine objects are created by: Figured_bass_engraver (page 421).

Standard settings:
    axes (list):
        '(1)
        List of axis numbers. In the case of alignment grobs, this should contain only one number.
    staff-staff-spacing (alist, with symbols as keys):
        '(((minimum-distance . 1.5) (padding . 0.1))
        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).

**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 #<procedure ly:axis-group-interface::height (_)> #<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): **axis-group-interface** (page 678), **grob-interface** (page 704), **outside-staff-axis-group-interface** (page 729), and **spanner-interface** (page 746).

This object is of class Spanner (characterized by **spanner-interface** (page 746)).

### 3.1.19 Beam

A beam.

Beam objects are created by: **Auto_beam_engraver** (page 402), **Beam_engraver** (page 407), **Chord_tremolo_engraver** (page 412), **Grace_auto_beam_engraver** (page 424), and **Grace_beam_engraver** (page 425).

Standard settings:

- **auto-knee-gap** (dimension, in staff space):
  
  5.5

  If a gap is found between note heads where a horizontal beam fits and it 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-accentual-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 (alist, with symbols as keys):
  '((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 \((\text{start} \ . \ \text{end})\), where \(\text{start}\) and \(\text{end}\) are vertical positions
    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 5575a25ef120 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1407:0
    (grob)>
    This makes the grob invisible.

vertical-skylines (pair of skylines):
    \#<unpure-pure-container \#<procedure ly:grob::vertical-skylines-from-
    stencil (_)> \#<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 \((\text{left} \ . \ \text{right})\), where both \text{left}
    and \text{right} are in staff-space units of the current staff.

This object supports the following interface(s): beam-interface (page 682),
    grob-interface (page 704), spanner-interface (page 746), staff-symbol-referencer-
    interface (page 749), and unbreakable-spanner-interface (page 762).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.20 BendAfter

A grob for displaying falls and doits.

BendAfter objects are created by: Bend_engraver (page 409).

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 \texttt{Staff.StaffSymbol.thickness}).

This object supports the following interface(s): \texttt{bend-after-interface} (page 685), \texttt{grob-interface} (page 704), and \texttt{spanner-interface} (page 746).

This object is of class \texttt{Spanner} (characterized by \texttt{spanner-interface} (page 746)).

### 3.1.21 BendSpanner

A string bending as used in tablature notation.

BendSpanner objects are created by: \texttt{BendSpannerEngraver} (page 410).

Standard settings:

\begin{itemize}
  \item \texttt{avoid-slur} (symbol):
  \begin{itemize}
    \item \texttt{ignore}
  \end{itemize}
  Method of handling slur collisions. Choices are \texttt{inside}, \texttt{outside}, \texttt{around}, and \texttt{ignore}. \texttt{inside} adjusts the slur if needed to keep the grob inside the slur. \texttt{outside} moves the grob vertically to the outside of the slur. \texttt{around} moves the grob vertically to the outside of the slur only if there is a collision. \texttt{ignore} does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), \texttt{outside} and \texttt{around} behave like \texttt{ignore}.
  \item \texttt{baseline-skip} (dimension, in staff space):
  \begin{itemize}
    \item 3
  \end{itemize}
  Distance between base lines of multiple lines of text.
  \item \texttt{before-line-breaking} (boolean):
  \begin{itemize}
    \item \texttt{bend::target-cautionary}
  \end{itemize}
  Dummy property, used to trigger a callback function.
  \item \texttt{details} (alist, with symbols as keys):
  \begin{itemize}
    \item \texttt{((arrow-stencil}
      \begin{itemize}
        \item \texttt{(#<procedure bend::arrow-head-stencil (thickness x-y-coords height width dir)>>)}
        \item \texttt{(curvature-factor . 0.35)}
        \item \texttt{(bend-arrowhead-height . 1.25)}
        \item \texttt{(bend-arrowhead-width . 0.8)}
      \end{itemize}
    \item \texttt{(bend-amount-strings}
      \begin{itemize}
        \item \texttt{(quarter . "1/4")}
        \item \texttt{(half . "1/2")}
        \item \texttt{(three-quarter . "3/4")}
        \item \texttt{(full . #f)}
      \end{itemize}
    \item \texttt{(curve-x-padding-line-end . 0.5)}
    \item \texttt{(curve-y-padding-line-end . 1)}
    \item \texttt{(dashed-line-settings 0.4 0.4 0)}
    \item \texttt{(head-text-break-visibility . #(#f #t #t))}
    \item \texttt{(horizontal-left-padding . 0.1)}
    \item \texttt{(successive-level . 1)}
    \item \texttt{(target-visibility . #f)}
    \item \texttt{(vertical-padding . 0.2)}
    \item \texttt{(y-distance-from-tabstaff-to-arrow-tip . 2.75))}
  \end{itemize}
\end{itemize}
A list 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):
  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):
  'latin1
  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-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.

padding (dimension, in staff space):
  0.15
  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.

spanner-id (index or symbol):
  ""
  An identifier to distinguish concurrent spanners.

stencil (stencil):
  bend-spanner::print
  The symbol to print.

style (symbol):
  '('
  This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

text (markup):
  #f
  Text markup. See Section “Formatting text” in Notation Reference.

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 \texttt{Staff.StaffSymbol.thickness}).

\texttt{vertical-skylines} (pair of skylines):
\begin{verbatim}
#<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-extents (_ _ _)> >
\end{verbatim}
Two skylines, one above and one below this grob.

\texttt{word-space} (dimension, in staff space):
\begin{verbatim}
0.6
\end{verbatim}
Space to insert between words in texts.

\texttt{Y-offset} (number):
\begin{verbatim}
0
\end{verbatim}
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): \texttt{bend-interface} (page 685), \texttt{font-interface} (page 698), \texttt{grob-interface} (page 704), \texttt{line-spanner-interface} (page 717), \texttt{outside-staff-interface} (page 730), \texttt{spanner-interface} (page 746), \texttt{text-interface} (page 755), and \texttt{text-script-interface} (page 756).

This object is of class \texttt{Spanner} (characterized by \texttt{spanner-interface} (page 746)).

3.1.22 \texttt{BreakAlignGroup}

An auxiliary grob to group several breakable items of the same type (clefs, time signatures, etc.) across staves so that they will be aligned horizontally. See also \texttt{BreakAlignment} (page 501).

\texttt{BreakAlignGroup} objects are created by: \texttt{Break_align_ engraver} (page 410).

Standard settings:

\texttt{axes} (list):
\begin{verbatim}
'(0)
\end{verbatim}
List of axis numbers. In the case of alignment grobs, this should contain only one number.

\texttt{break-align-anchor} (number):
\begin{verbatim}
ly:break-aligned-interface::calc-average-anchor
\end{verbatim}
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.

\texttt{break-align-anchor-alignment} (number):
\begin{verbatim}
ly:break-aligned-interface::calc-joint-anchor-alignment
\end{verbatim}
Read by \texttt{ly:break-aligned-interface::calc-extent-aligned-anchor} for aligning an anchor to a grob’s extent.

\texttt{break-visibility} (vector):
\begin{verbatim}
ly:break-aligned-interface::calc-break-visibility
\end{verbatim}
A vector of 3 booleans, \#(\texttt{end-of-line unbroken begin-of-line}). \#t means visible, \#f means killed.
X-extent (pair of numbers):
\[\text{ly:axis-group-interface::width}\]
Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): axis-group-interface (page 678), break-aligned-interface (page 687), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.23 BreakAlignment

An auxiliary grob that manages the horizontal ordering of BreakAlignGroup (page 500), grobs within a NonMusicalPaperColumn (page 593), grob (for example, whether the time signature follows or precedes a bar line).

BreakAlignment objects are created by: Break_align_engraver (page 410).

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
  staff-ellipsis
  cue-end-clef
  ambitus
  breathing-sign
  signum-repetitionis
  clef
  cue-clef
  staff-bar
  key-cancellation
  key-signature
  time-signature
  custos)
(left-edge
  staff-ellipsis
  cue-end-clef
  ambitus
  breathing-sign
  signum-repetitionis
  clef
  cue-clef
  staff-bar
  key-cancellation
  key-signature
  time-signature
  custos)
(left-edge
  staff-ellipsis
  cue-end-clef
  ambitus
  breathing-sign
  signum-repetitionis
  clef
  cue-clef
  staff-bar
  key-cancellation
  key-signature
  time-signature
  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): axis-group-interface (page 678),
break-alignment-interface (page 688), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.24 BreathingSign

A breathing sign.

BreathingSign objects are created by: Breathing_sign_engraver (page 410), and
Caesura_engraver (page 410).

Standard settings:

break-align-symbol (symbol):
'brespiration
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.

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.

non-musical (boolean):
  #t
  True if the grob belongs to a NonMusicalPaperColumn.

space-alist (alist, with symbols as keys):
  '((ambitus extra-space . 2.0)
   (custos minimum-space . 1.0)
   (key-signature minimum-space . 1.5)
   (time-signature minimum-space . 1.5)
   (signum-repetitionis 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 [Wansk e] page 126–134, [Ross] page 143–147.

stencil (stencil):
    ly:text-interface::print
    The symbol to print.

thickness (number):
    1.9
    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 #<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 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): break-aligned-interface (page 687), breathing-sign-interface (page 689), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.25 CaesuraScript
A script for \caesura, e.g., an outside-staff comma or a fermata over a bar line.

CaesuraScript objects are created by: Caesura_ engraver (page 410).
Standard settings:

before-line-breaking (boolean):
    caesura-script-interface::before-line-breaking
    Dummy property, used to trigger a callback function.

break-visibility (vector):
    #(#t #t #f)
    A vector of 3 booleans, #(end-of-line unbroken begin-of-line). #t means visible,
    #f means killed.

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.

non-musical (boolean):
    #t
    True if the grob belongs to a NonMusicalPaperColumn.

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 "<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 "<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> "<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)>
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): caesura-script-interface (page 690),
font-interface (page 698), grob-interface (page 704), item-interface (page 712),
outside-staff-interface (page 730), script-interface (page 734), self-alignment-interface (page 736), and side-position-interface (page 738).

This object is of class Item (characterized by item-interface (page 712)).

3.1.26 CenteredBarNumber
A centered bar number; see also CenteredBarNumberLineSpanner (page 507). Ordinary bar numbers are managed with BarNumber (page 489), grobs.

CenteredBarNumber objects are created by: Bar_number_engraver (page 406).

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).

font-family (symbol):
   'roman
   The font family is the broadest category for selecting text fonts. Options include: sans, roman.

font-size (number):
   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.

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.

X-offset (number):
  centered-spanner-interface::calc-x-offset
  The horizontal amount that this object is moved relative to its X-parent.

This object supports the following interface(s): bar-number-interface (page 682), centered-bar-number-interface (page 690), centered-spanner-interface (page 690), font-interface (page 698), grob-interface (page 704), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.27 CenteredBarNumberLineSpanner
An auxiliary grob providing a vertical baseline to align CenteredBarNumber (page 506), grobs.

CenteredBarNumberLineSpanner objects are created by: Centered_bar_number_align_ engraver (page 411).

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.

outside-staff-priority (number):
  1200
  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):
  4
  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 #<procedure ly:grob::vertical-skylines-from-element-stencils (_)> #<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):
  ly:axis-group-interface::height (_)
  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):
  ly:side-position-interface::y-aligned-side (_ #:optional _)
  ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): axis-group-interface (page 678), bar-number-interface (page 682), centered-bar-number-line-spanner-interface (page 690), grob-interface (page 704), outside-staff-interface (page 730), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.28 ChordName

A stand-alone chord name. For chord names in chord grids, see GridChordName (page 553).

ChordName objects are created by: Chord_name_engraver (page 412).

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):
  #$<unpure-pure-container #$<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): accidental-switch-interface (page 676), chord-name-interface (page 691), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), rhythmic-grob-interface (page 734), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.29 ChordSquare

In a chord grid, this grob represents one chord square. It helps place GridChordName (page 553), grobs, and draws lines to separate them. Note that this grob only draws the diagonal lines in a square. The borders of the square are drawn by StaffSymbol (page 632), and BarLine (page 486).

ChordSquare objects are created by: Chord_square_engraver (page 412).

Standard settings:

  measure-division-chord-placement-alist (association list (list of pairs)):
  '(((1) (0 . 0))
   ((1/2  1/2) (-0.4 0.4) (0.4 -0.4))
   ((1/2  1/4 1/4) (-0.4 0.4) (0 . -0.65) (0.63 . 0))
   ((1/4  1/4 1/2) (-0.63 0) (0 . 0.65) (0.4 . -0.4))
   ((1/4  1/4 1/4 1/4) (-0.63 0) (0 . 0.7)
An alist mapping measure divisions (see the measure-division property) to lists of coordinates (number pairs) applied to the chord names of a chord square. Coordinates are normalized between -1 and 1 within the square.

measure-division-lines-alist (association list (list of pairs)):
'(((1))
  ((1/2 1/2) (-1 -1 1 1))
  ((1/2 1/4 1/4) (-1 -1 1 1) (0 0 1 -1))
  ((1/4 1/4 1/2) (-1 -1 1 1) (-1 1 0 0))
  ((1/4 1/4 1/4 1/4) (-1 -1 1 1) (-1 1 1 -1))
  ((1/4 3/4) (-1 -1 0 0) (-1 1 0 0))
  ((3/4 1/4) (0 0 1 -1) (0 0 1 1)))

An alist mapping measure divisions (see the measure-division property) to lists of lines to draw in the square, given as 4-element lists: (x-start y-start x-end y-end).

stencil (stencil):
chord-square::print
The symbol to print.

X-extent (pair of numbers):
chord-square::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 #<procedure chord-square::height (grob)>>
Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): chord-square-interface (page 691), grob-interface (page 704), line-interface (page 716), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.30 Clef
A clef. See also ClefModifier (page 513), CueClef (page 521), and CueEndClef (page 524).

Clef objects are created by: Clef engraver (page 413).

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 (alist, with symbols as keys):
  '((cue-clef extra-space . 2.0)
    (signum-repetitionis extra-space . 0.7)
    (staff-bar extra-space . 0.7)
    (ambitus extra-space . 1.15)
    (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 #<procedure ly:grob::vertical-skylines-from-stencil (_)>>
Two skylines, one above and one below this grob.

Y-extent (pair of numbers):
#<unpure-pure-container #<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):

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-aligned-interface (page 687), clef-interface (page 691), font-interface (page 698), grob-interface (page 704), item-interface (page 712), pure-from-neighbor-interface (page 733), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.31 ClefModifier

A grob that draws the clef modifier (if present), in most cases the digit 8 below or above the clef. See also Clef (page 510), CueClef (page 521), and CueEndClef (page 524).

ClefModifier objects are created by: Clef_engraver (page 413), and Cue_clef_engraver (page 415).

Standard settings:

break-visibility (vector):

A vector of 3 booleans, #(end-of-line unbroken begin-of-line). #t means visible, #f means killed.

clef-alignments (alist, with symbols as keys):

An alist of parent-alignments that should be used for clef modifiers with various clefs

color (color):

The color of this grob.

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.

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.

**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):

\[
\text{ly:text-interface::print}
\]

The symbol to print.

**transparent** (boolean):

\[
\text{#<procedure 5575a25ef090 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1407:0 (grob)>}
\]

This makes the grob invisible.

**vertical-skylines** (pair of skylines):

\[
\text{#<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-stencil (_)> >}
\]

Two skylines, one above and one below this grob.

**X-offset** (number):

\[
\text{ly:self-alignment-interface::aligned-on-x-parent}
\]

The horizontal amount that this object is moved relative to its X-parent.

**Y-extent** (pair of numbers):

\[
\text{#<unpure-pure-container #<procedure ly:grob::stencil-height (_)> >}
\]

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

**Y-offset** (number):

\[
\text{#<unpure-pure-container #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _) > #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >}
\]

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): clef-modifier-interface (page 692), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.32 ClusterSpanner

A cluster spanner. The envelope shape within the spanner is given by ClusterSpannerBeacon (page 515), grobs.

ClusterSpanner objects are created by: ClusterSpanner_ engraver (page 413).

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): cluster-interface (page 692),
grob-interface (page 704), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.33 ClusterSpannerBeacon
An auxiliary grob to specify the minimum and maximum pitch of a ClusterSpanner (page 514),
grob at a given moment.

ClusterSpannerBeacon objects are created by: Cluster_spanner_engraver (page 413).

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): cluster-beacon-interface (page 692),
grob-interface (page 704), item-interface (page 712), and rhythmic-grob-interface
(page 734).

This object is of class Item (characterized by item-interface (page 712)).

3.1.34 CodaMark
A coda mark.

CodaMark objects are created by: Mark_engraver (page 432).

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):
  #(t t f)
  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-padding (number):
  0.4
  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):
  1400
  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):
  0.4
  Add this much extra space between objects that are next to each other.
self-alignment-X (number):
  break-alignable-interface::self-alignment-opposite-of-anchor
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 #<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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-alignable-interface (page 687), coda-mark-interface (page 692), font-interface (page 698), grob-interface (page 704), item-interface (page 712), mark-interface (page 720), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.35 CombineTextScript
A grob for printing markup given in the soloText, soloIIText, and aDueText properties if automatic part combining is active.

CombineTextScript objects are created by: Part_combine_engraver (page 441).

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):
#f
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):
#f
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 #<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 ly:side-position-interface::y-
   aligned-side (_ #:optional _) #<procedure ly:side-position-
   interface::pure-y-aligned-side (_ _ #:optional _) > >
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface
(page 676), font-interface (page 698), grob-interface (page 704), item-interface
(page 712), outside-staff-interface (page 730), self-alignment-interface
(page 736), side-position-interface (page 738), text-interface (page 755), and
text-script-interface (page 756).

This object is of class Item (characterized by item-interface (page 712)).

3.1.36 ControlPoint

A visual representation of a Bézier control point in ties and slurs.

ControlPoint objects are created by: Show_control_points_engraver (page 446).

Standard settings:

   color (color):
      "IndianRed"
      The color of this grob.

   horizontal-skylines (pair of skylines):
      #f
      Two skylines, one to the left and one to the right of this grob.

   layer (integer):
      3
      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.

stencil (stencil):
   ly:text-interface::print
   The symbol to print.

text (markup):
   '(#<procedure draw-circle-markup (layout props radius thickness filled)>
      0.3
      0.01
Text markup. See Section “Formatting text” in Notation Reference.

vertical-skylines (pair of skylines):

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.

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.

This object supports the following interface(s): control-point-interface (page 693), grob-interface (page 704), sticky-grob-interface (page 752), and text-interface (page 755).

This object can be of either of the following classes: Item (characterized by item-interface) or Spanner (characterized by spanner-interface). It supports the following interfaces conditionally depending on the class: item-interface (page 712), and spanner-interface (page 746).

3.1.37 ControlPolygon

A visual representation of a Bézier control polygon as used in ties and slurs.

ControlPolygon objects are created by: Show_control_points_engraver (page 446).

Standard settings:

```
color (color):
  "BurlyWood"
  The color of this grob.
```

extroversion (number):

0.5

For polygons, how the thickness of the line is spread on each side of the exact polygon with ideal zero thickness. If this is 0, the middle of line is on the polygon. If 1, the line sticks out of the polygon. If -1, the outer side of the line is exactly on the polygon. Other numeric values are interpolated.

filled (boolean):

#f

Whether an object is filled with ink.
horizontal-skylines (pair of skylines):
  #f
  Two skylines, one to the left and one to the right of this grob.

layer (integer):
  2
  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.

stencil (stencil):
  ly:text-interface::print
  The symbol to print.

text (markup):
  control-polygon::calc-text
  Text markup. See Section “Formatting text” in Notation Reference.

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):
  #f
  Two skylines, one above and one below this grob.

X-extent (pair of numbers):
  #f
  Extent (size) in the X direction, measured in staff-space units, relative to object’s
  reference point.

Y-extent (pair of numbers):
  #f
  Extent (size) in the Y direction, measured in staff-space units, relative to object’s
  reference point.

This object supports the following interface(s): control-polygon-interface (page 693),
  grob-interface (page 704), sticky-grob-interface (page 752), and text-interface
  (page 755).

This object can be of either of the following classes: Item (characterized by item-interface)
  or Spanner (characterized by spanner-interface). It supports the following interfaces condi-
  tionally depending on the class: item-interface (page 712), and spanner-interface
  (page 746).

3.1.38 CueClef
A clef starting a cue. See also Clef (page 510), ClefModifier (page 513), and CueEndClef
  (page 524).

CueClef objects are created by: Cue_clef_ engraver (page 415).
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.
Chapter 3: Backend

space-alist (alist, with symbols as keys):
'((signum-repetitionis minimum-space . 2.7)
 (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 #<procedure ly:grob::vertical-skylines-from-stencil (_)>>
  Two skylines, one above and one below this grob.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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 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): break-aligned-interface (page 687), clef-interface (page 691), font-interface (page 698), grob-interface (page 704), item-interface (page 712), pure-from-neighbor-interface (page 733), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.39 CueEndClef

A clef ending a cue. See also Clef (page 510), ClefModifier (page 513), and CueClef (page 521).

CueEndClef objects are created by: Cue_clef_engraver (page 415).

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 (alist, with symbols as keys):
    '(((clef extra-space . 0.7)
        (cue-clef extra-space . 0.7)
        (signum-repetitionis 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

stencil (stencil):
  ly:clef::print
  The symbol to print.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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 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): break-aligned-interface (page 687),
clef-interface (page 691), font-interface (page 698), grob-interface (page 704),
item-interface (page 712), pure-from-neighbor-interface (page 733), and
staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).
3.1.40 Custos

A custos, mainly used in older notation like Gregorian chant.

Custos objects are created by: Custos_engraver (page 417).

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 (alist, with symbols as keys):
  '((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

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 #<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): break-aligned-interface (page 687),
custos-interface (page 693), font-interface (page 698), grob-interface (page 704),
item-interface (page 712), and staff-symbol-referencer-interface (page 749).
This object is of class Item (characterized by item-interface (page 712)).

3.1.41 Divisio
A structural divider in a chant, often calling for a breath or caesura.

Divisio objects are created by: Divisio_engraver (page 417).

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-anchor-alignment (number):
0
Read by ly:break-aligned-interface::calc-extent-aligned-anchor for align-
ing an anchor to a grob’s extent.

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.

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-height (pair of numbers):
  item::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):
  '(-1.0 . 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).

non-musical (boolean):
  #t
  True if the grob belongs to a NonMusicalPaperColumn.

space-alist (alist, with symbols as keys):
  '((ambitus extra-space . 1.0)
   (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 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

stencil (stencil):
   ly:text-interface::print
   The symbol to print.

thickness (number):
   1.9
   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 #<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 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): break-aligned-interface (page 687), breathing-sign-interface (page 689), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.42 DotColumn

An auxiliary grob to align stacked Dots (page 531), grobs of dotted notes and chords.

DotColumn objects are created by: Dot_column_engraver (page 418), and Vaticana_ligature_engraver (page 456).

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): axis-group-interface (page 678), dot-column-interface (page 693), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.43 Dots

The dot(s) of a dotted note. See also DotColumn (page 531).

Dots objects are created by: Dots_engraver (page 418).

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
  \((-\text{inf.} . 0 . +\text{inf.} 0)\).

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
  \((+\text{inf.} 0 . -\text{inf.} 0)\).

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 #<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): dots-interface (page 694),
font-interface (page 698), grob-interface (page 704), item-interface (page 712), and
staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.44 DoublePercentRepeat

A double-percent symbol for repeating two bars. See also DoublePercentRepeatCounter
(page 533), PercentRepeat (page 602), DoubleRepeatSlash (page 535), and RepeatSlash
(page 609).

DoublePercentRepeat objects are created by: Double_percent_repeat_engraver
(page 418).

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-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 #<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): break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), and percent-repeat-interface (page 731).

This object is of class Item (characterized by item-interface (page 712)).

3.1.45 DoublePercentRepeatCounter
A grob to print a counter for DoublePercentRepeat (page 532), grobs.

DoublePercentRepeatCounter objects are created by: Double_percent_repeat_engraver (page 418).

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-features` (list):

'("cv47")'

Opentype features.

`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.
Chapter 3: Backend

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.

This object supports the following interface(s): font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.46 DoubleRepeatSlash

A double-percent symbol for repeating patterns shorter than a single measure, and which contain mixed durations. See also PercentRepeat (page 602), DoublePercentRepeat (page 532), and RepeatSlash (page 609).

DoubleRepeatSlash objects are created by: Slash_repeat_engraver (page 446).

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-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):

Y-extent is the size in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), percent-repeat-interface (page 731), and rhythmic-grob-interface (page 734).

This object is of class Item (characterized by item-interface (page 712)).

3.1.47 DurationLine

A horizontal duration line, continuing rhythmic items (usually note heads).

DurationLine objects are created by: Duration_line_engraver (page 419).

Standard settings:

- **after-line-breaking** (boolean):
  - `ly:spanner::kill-zero-spanned-time`
    Dummy property, used to trigger callback for after-line-breaking.

- **arrow-length** (number):
  - 2
    Arrow length.

- **arrow-width** (number):
  - 1.5
    Arrow width.

- **bound-details** (alist, with symbols as keys):
  - `'(right (attach-dir . -1)
      (end-on-accidental . #t)
      (end-on-arpeggio . #t)
      (padding . 0.4)
      (end-style . #f))`
    (right-broken (padding . 0.4) (end-style . #f))
    (left-broken (padding . 0.5))
    (left (attach-dir . 1)
      (padding . -0.3)
      (start-at-dot . #f))`
  An alist of properties for determining attachments of spanners to edges.

- **breakable** (boolean):
  - #t
    Allow breaks here.

- **details** (alist, with symbols as keys):
  - `'( (hook-height . 0.34)
      (hook-thickness . #f)
      (hook-direction . 1)
      (extra-dot-padding . 0.5))`
  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.
left-bound-info (alist, with symbols as keys):
   ly:horizontal-line-spanner::calc-left-bound-info
   An alist of properties for determining attachments of spanners to edges.

minimum-length (dimension, in staff space):
   2
   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):
   6
   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.

right-bound-info (alist, with symbols as keys):
   ly:horizontal-line-spanner::calc-right-bound-info
   An alist of properties for determining attachments of spanners to edges.

springs-and-rods (boolean):
   ly:spanner::set-spacing-rods
   Dummy variable for triggering spacing routines.

stencil (stencil):
   duration-line::print
   The symbol to print.

style (symbol):
   'beam
   This setting determines in what style a grob is typeset. Valid choices depend on the
   stencil callback reading this property.

thickness (number):
   4
   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):
   #f
   If true, the spanner will stop at the bar line just before it would otherwise stop.

vertical-skylines (pair of skylines):
   #<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-
   stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-
   extents (_ _ _)> >
   Two skylines, one above and one below this grob.

Y-offset (number):
   0
   The vertical amount that this object is moved relative to its Y-parent.
zigzag-length (dimension, in staff space):
1
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):
1
The width of one zigzag squiggle. This number is adjusted slightly so that the spanner line can be constructed from a whole number of squiggles.

This object supports the following interface(s): duration-line-interface (page 694), font-interface (page 698), grob-interface (page 704), horizontal-line-spanner-interface (page 710), line-interface (page 716), line-spanner-interface (page 717), spanner-interface (page 746), and unbreakable-spanner-interface (page 762).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.48 DynamicLineSpanner
An auxiliary grob providing a vertical baseline to align successive dynamic grobs (DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555)) within a staff.

DynamicLineSpanner objects are created by: Dynamic_align_engraver (page 419).

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.
Chapter 3: Backend

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):

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):

ly:axis-group-interface::height

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.

3.1.49 DynamicText

A dynamic text item like ‘ff’ or ‘mp’. See also DynamicLineSpanner (page 538).

DynamicText objects are created by: Dynamic_ engraver (page 420).

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 \((+\infty, 0 . -\infty, 0)\).

**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 #<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 #<procedure ly:grob::stencil-height (_)> >

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): dynamic-interface (page 695), dynamic-text-interface (page 695), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), script-interface (page 734), self-alignment-interface (page 736), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.50 DynamicTextSpanner

Dynamic text like ‘cresc’, usually followed by a (dashed) line. See also DynamicLineSpanner (page 538), and TextSpanner (page 652).

DynamicTextSpanner objects are created by: Dynamic_ engraver (page 420).

Standard settings:

- `before-line-breaking` (boolean):
  - dynamic-text-spanner::before-line-breaking
    
    Dummy property, used to trigger a callback function.

- `bound-details` (alist, with symbols as keys):
  - '((right (attach-dir . -1) (padding . 0.75))
    (right-broken (attach-dir . 1) (padding . 0.0)))
  - (left (attach-dir . -1)
    (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` (alist, with symbols as keys):
  - ly:horizontal-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 (alist, with symbols as keys):
   ly:horizontal-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 #<procedure ly:grob::vertical-skylines-from-
   stencil (_)> #<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): dynamic-interface (page 695),
   dynamic-text-spanner-interface (page 695), font-interface (page 698), grob-interface
   (page 704), horizontal-line-spanner-interface (page 710), line-interface (page 716),
   line-spanner-interface (page 717), spanner-interface (page 746), and text-interface
   (page 755).

   This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.51 Episema

An episema line (over a group of notes). Used in Gregorian chant.

Episema objects are created by: Episema_engraver (page 421).

Standard settings:

   bound-details (alist, with symbols as keys):
   '(((left (padding . 0) (attach-dir . -1))
(right (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 (alist, with symbols as keys):**

`ly:horizontal-line-spanner::calc-left-bound-info`

An alist of properties for determining attachments of spanners to edges.

**right-bound-info (alist, with symbols as keys):**

`ly:horizontal-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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _) >

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): episema-interface (page 696), font-interface (page 698), grob-interface (page 704), horizontal-line-spanner-interface (page 710), line-interface (page 716), line-spanner-interface (page 717), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.52 FingerGlideSpanner

A line connecting two Fingering (page 545), grobs, usually indicating a gliding finger for stringed instruments.

FingerGlideSpanner objects are created by: Finger_glide_engraver (page 422).

Standard settings:

**bound-details (alist, with symbols as keys):**

'((right (attach-dir . -1)
  (right-stub-length . 1)
  (padding . 0.2))
(left (attach-dir . 1)
  (left-stub-length . 1)
  (padding . 0.2)))

An alist of properties for determining attachments of spanners to edges.

dash-fraction (number):
  0.4
  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):
  1
  The length of one dash together with whitespace. If negative, no line is drawn at all.

details (alist, with symbols as keys):
  '((bow-direction . #f))
  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.

left-bound-info (alist, with symbols as keys):
  ly:line-spanner::calc-left-bound-info
  An alist of properties for determining attachments of spanners to edges.

minimum-length (dimension, in staff space):
  2.5
  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.

minimum-length-after-break (dimension, in staff space):
  2.5
  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):
  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 (alist, with symbols as keys):
  ly:line-spanner::calc-right-bound-info
  An alist of properties for determining attachments of spanners to edges.

springs-and-rods (boolean):
  ly:spanner::set-spacing-rods
  Dummy variable for triggering spacing routines.

stencil (stencil):
  finger-glide::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):
  1.4
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 #<procedure ly:grob::vertical-skylines-from-
  stencil (_) #<procedure ly:grob::pure-simple-vertical-skylines-from-
  extents (_ _ _)_>_>
  Two skylines, one above and one below this grob.

zigzag-length (dimension, in staff space):
  1
  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):
  1
  The width of one zigzag squiggle. This number is adjusted slightly so that the spanner
  line can be constructed from a whole number of squiggles.

This object supports the following interface(s): finger-glide-interface (page 696),
grob-interface (page 704), line-spanner-interface (page 717), and spanner-interface
(page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.53 Fingering

A fingering symbol (usually a digit). See also FingeringColumn (page 547), and StrokeFinger
(page 640).

Fingering objects are created by: Fingering_engraver (page 422), and New_fingering_engraver (page 437).

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-features (list):
  '("cv47" "ss01")
  Opentype features.

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 #<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): finger-interface (page 698),
    font-interface (page 698), grob-interface (page 704), item-interface (page 712),
    outside-staff-interface (page 730), self-alignment-interface (page 736),
    side-position-interface (page 738), text-interface (page 755), and text-script-
    interface (page 756).

This object is of class Item (characterized by item-interface (page 712)).

3.1.54 FingeringColumn

An auxiliary grob to align stacked Fingering (page 545), grobs.

FingeringColumn objects are created by: Fingering_column_engraver (page 422).

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): finger-column-interface (page 698),
    grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.55 Flag

A flag (in the musical sense).

Flag objects are created by: Stem_engraver (page 450).

Standard settings:

    color (color):
    #<procedure 5575a25ef030 at /build/out/share/lilypond/current/scm/lily/output-lib.scm>
    (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 5575a25ef060 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1407:0
  (grob)>
  This makes the grob invisible.

vertical-skylines (pair of skylines):
  #<unpure-pure-container #<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 #<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 ly:flag::calc-y-offset (_)> >
  #<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): flag-interface (page 698),
font-interface (page 698), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.56 Footnote

A footnote mark (usually a number) with a pointing line attached to another grob.

Footnote objects are created by: Footnote_ engraver (page 423).

Standard settings:

after-line-breaking (boolean):
  ly:balloon-interface::remove-irrelevant-spanner
  Dummy property, used to trigger callback for after-line-breaking.

annotation-balloon (boolean):
  #f
  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 5575a26ab820 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  If set, footnotes are automatically numbered.

break-visibility (vector):
  #<procedure 5575a26ab800 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:2973:0 (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 5575a26ab7c0 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  A footnote for the grob.

stencil (stencil):
  ly:balloon-interface::print
  The symbol to print.

text (markup):
  #<procedure 5575a26ab7a0 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  Text markup. See Section “Formatting text” in Notation Reference.

X-extent (pair of numbers):
  #f
  Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

X-offset (number):
  #<procedure 5575a26ab780 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  The horizontal amount that this object is moved relative to its X-parent.

Y-extent (pair of numbers):
  #f
  Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

Y-offset (number):
  #<procedure 5575a26ab760 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): balloon-interface (page 680),
font-interface (page 698), footnote-interface (page 700), grob-interface (page 704),
sticky-grob-interface (page 752), and text-interface (page 755).
This object can be of either of the following classes: Item (characterized by item-interface) or Spanner (characterized by spanner-interface). It supports the following interfaces conditionally depending on the class: item-interface (page 712), and spanner-interface (page 746).

3.1.57 FretBoard

A fretboard diagram.

FretBoard objects are created by: Fretboard_engraver (page 423).

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 (alist, with symbols as keys):
  '((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.

• **handedness** – Print the fret-diagram left- or right-handed. -1, LEFT for left ; 1, RIGHT for right. Default RIGHT.

• **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 arabic, roman-ij-lower, roman-ij-upper, roman-lower, roman-upper, arabic and custom. In the last 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 \( \text{thickness} \ast (1+\text{string-thickness-factor})^\ast (k-1) \). 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 #<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): chord-name-interface (page 691), font-interface (page 698), fret-diagram-interface (page 700), grob-interface
This object is of class Item (characterized by item-interface).

### 3.1.58 Glissando

A glissando line.

Glissando objects are created by: Glissando_ engraver (page 424).

Standard settings:

- **after-line-breaking** (boolean):
  - ly:spanner::kill-zero-spanned-time
    - Dummy property, used to trigger callback for after-line-breaking.

- **bound-details** (alist, with symbols as keys):
  - `(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** (alist, with symbols as keys):
  - 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** (alist, with symbols as keys):
  - 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.

- **vertical-skylines** (pair of skylines):
  - `(unpure-pure-container (procedure ly:grob::vertical-skylines-from-stencil (_)) (procedure ly:grob::pure-simple-vertical-skylines-from-extents (_ _)))`

  - Two skylines, one above and one below this grob.
Chapter 3: Backend

**3.1.59 GraceSpacing**

An auxiliary grob to handle (horizontal) spacing of grace notes. See also NoteSpacing (page 598), StaffSpacing (page 632), and SpacingSpanner (page 626).

GraceSpacing objects are created by: Grace_spacing_engraver (page 425).

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): grace-spacing-interface (page 702), grob-interface (page 704), spacing-options-interface (page 744), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

**3.1.60 GridChordName**

A chord name in a chord grid.

GridChordName objects are created by: Grid_chord_name_engraver (page 425).

Standard settings:

- **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.

X-offset (number):
   #<procedure 5575a211a400 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:3135:0 (grob)>
   The horizontal amount that this object is moved relative to its X-parent.

Y-offset (number):
   #<procedure 5575a211a320 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:3135:0 (grob)>
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface (page 676), font-interface (page 698), grid-chord-name-interface (page 703), grob-interface (page 704), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.61 GridLine

A vertical line between staves, indicating rhythmic synchronization. See also GridPoint (page 555).

GridLine objects are created by: Grid_line_span_engraver (page 426).

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): grid-line-interface (page 703),
grob-interface (page 704), item-interface (page 712), and self-alignment-interface
(page 736).

This object is of class Item (characterized by item-interface (page 712)).

3.1.62 GridPoint

An auxiliary grob marking a start or end point for a GridLine (page 554), grob.
GridPoint objects are created by: Grid_point_engraver (page 426).

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): grid-point-interface (page 703),
grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.63 Hairpin

A hairpin. See also DynamicLineSpanner (page 538).

Hairpin objects are created by: Dynamic_engraver (page 420).

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):
   #f
   Put a circle at start/end of hairpins (al/del niente).
endpoint-alignments (pair of numbers):
    '(-1 . 1)
A pair of numbers representing the alignments of an object’s endpoints. E.g., the
ends of a hairpin relative to NoteColumn grobs.

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 #<procedure ly:grob::vertical-skylines-from-
    stencil (_) #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 #procedure ly:grob::stencil-height (_) #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 #<procedure ly:self-alignment-interface::y-aligned-on-self (_)> #<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): dynamic-interface (page 695),
grob-interface (page 704), hairpin-interface (page 708), line-interface (page 716),
outside-staff-interface (page 730), self-alignment-interface (page 736), and
spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.64 HorizontalBracket

A horizontal bracket between notes. See also HorizontalBracketText (page 558), and
MeasureSpanner (page 583).

HorizontalBracket objects are created by: Horizontal_bracket_engraver (page 427).

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):**

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): grob-interface (page 704), horizontal-bracket-interface (page 709), line-interface (page 716), outside-staff-interface (page 730), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.65 *HorizontalBracketText*

Text (markup) for a HorizontalBracket (page 557), grob.

*HorizontalBracketText* objects are created by: HorizontalBracket_ engraver (page 427).

**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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), horizontal-bracket-text-interface (page 710), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.66 InstrumentName
An instrument name, usually displayed to the left of a staff.

InstrumentName objects are created by: Instrument_name_engraver (page 427).

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):

\texttt{system-start-text::calc-x-offset}

The horizontal amount that this object is moved relative to its X-parent.

Y-offset (number):

\texttt{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): \texttt{accidental-switch-interface} (page 676), \texttt{font-interface} (page 698), \texttt{grob-interface} (page 704), \texttt{self-alignment-interface} (page 736), \texttt{side-position-interface} (page 738), \texttt{spanner-interface} (page 746), \texttt{system-start-text-interface} (page 754), and \texttt{text-interface} (page 755).

This object is of class Spanner (characterized by \texttt{spanner-interface} (page 746)).

\subsection{3.1.67 InstrumentSwitch}

This grob is deprecated. Do not use it.

\texttt{InstrumentSwitch} objects are created by: \texttt{Instrument_switch_engraver} (page 427).

Standard settings:

\begin{itemize}
\item \texttt{direction (direction): 1}
\begin{itemize}
\item If \texttt{side-axis} is 0 (or X), then this property determines whether the object is placed \texttt{LEFT}, \texttt{CENTER} or \texttt{RIGHT} with respect to the other object. Otherwise, it determines whether the object is placed \texttt{UP}, \texttt{CENTER} or \texttt{DOWN}. Numerical values may also be used: \texttt{UP}=1, \texttt{DOWN}=-1, \texttt{LEFT}=-1, \texttt{RIGHT}=1, \texttt{CENTER}=0.
\end{itemize}
\item \texttt{extra-spacing-width (pair of numbers):'}\texttt{(+inf.0 . -inf.0)}
\begin{itemize}
\item 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 \texttt{(+inf.0 . -inf.0)}.
\end{itemize}
\item \texttt{outside-staff-priority (number): 500}
\begin{itemize}
\item 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 \texttt{outside-staff-priority} is closer to the staff.
\end{itemize}
\item \texttt{padding (dimension, in staff space): 0.5}
\begin{itemize}
\item Add this much extra space between objects that are next to each other.
\end{itemize}
\item \texttt{parent-alignment-X (number):} \texttt{#f}
\begin{itemize}
\item Specify on which point of the parent the object is aligned. The value \texttt{-1} means aligned on parent’s left edge, \texttt{0} on center, and \texttt{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 \texttt{self-alignment-X} property will be used.
\end{itemize}
\item \texttt{self-alignment-X (number): -1}
\begin{itemize}
\item Specify alignment of an object. The value \texttt{-1} means left aligned, \texttt{0} centered, and \texttt{1} right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.
\end{itemize}
\end{itemize}
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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.68 JumpScript

A grob to display a ‘point of departure’ like D.C. al fine.

JumpScript objects are created by: Jump_engraver (page 428).

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):
  #(t t f)
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: \( \text{UP}=1, \text{DOWN}=-1, \text{LEFT}=-1, \text{RIGHT}=1, \text{CENTER}=0 \).

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.

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 it is 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):
  1350
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 \( \text{outside-staff-priority} \) 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.

stencil (stencil):
  ly:text-interface::print
The symbol to print.

vertical-skylines (pair of skylines):
  #<unpure-pure-container #<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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), jump-script-interface (page 714), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.69 KeyCancellation

A key cancellation, normally consisting of naturals, to be displayed (if necessary) immediately before a KeySignature (page 565), grob if the key changes.

KeyCancellation objects are created by: Key_engraver (page 429).

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
genereal 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.

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
genereal 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 (alist, with symbols as keys):
  '(
    (time-signature extra-space . 1.25)
    (signum-repetitionis extra-space . 0.6)
    (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 #<procedure ly:grob::vertical-skylines-from-stencil (_)> >

Two skylines, one above and one below this grob.

Y-extent (pair of numbers):

#<unpure-pure-container #<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 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): accidental-switch-interface (page 676), break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), key-cancellation-interface (page 714), key-signature-interface (page 714), pure-from-neighbor-interface (page 733), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.70 KeySignature

A key signature. See also KeyCancellation (page 563).

KeySignature objects are created by: Key_engraver (page 429).

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.

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 (alist, with symbols as keys):
  '((ambitus extra-space . 1.15)
   (time-signature extra-space . 1.15)
   (signum-repetitionis extra-space . 1.1)
   (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 #<procedure ly: grob::vertical-skylines-from-stencil (._)> >
  Two skylines, one above and one below this grob.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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 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): accidental-switch-interface (page 676), break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), key-signature-interface (page 714), pure-from-neighbor-interface (page 733), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.71 KievanLigature

An auxiliary grob to handle a melisma (ligature) as used in Kievan square notation. See also MensuralLigature (page 585), VaticanaLigature (page 667), and LigatureBracket (page 573).

KievanLigature objects are created by: Kievan_ligature_engraver (page 430).

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): font-interface (page 698), grob-interface (page 704), kievan-ligature-interface (page 715), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).
3.1.72 LaissezVibrerTie

A laissez-vibrer tie (i.e., a tie from a note into nothing). See also LaissezVibrerTieColumn (page 570), RepeatTie (page 610), and Tie (page 653).

LaissezVibrerTie objects are created by: Laissez_vibrer_engraver (page 430).

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 (alist, with symbols as keys):
  - '((ratio . 0.333) (height-limit . 1.0))
    - A list 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 #<procedure ly:grob::vertical-skylines-from-stencil (_) > >
    - Two skylines, one above and one below this grob.
Y-extent (pair of numbers):
   #<unpure-pure-container #<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): bezier-curve-interface (page 687),
grob-interface (page 704), item-interface (page 712), semi-tie-interface (page 737), and tie-interface (page 757).

This object is of class Item (characterized by item-interface (page 712)).

3.1.73 LaissezVibrerTieColumn
An auxiliary grob to determine direction and shape of stacked LaissezVibrerTie (page 569), grobs.
LaissezVibrerTieColumn objects are created by: Laissez_vibrer_engraver (page 430).
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):
      #f
      Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.
   Y-extent (pair of numbers):
      #f
      Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): grob-interface (page 704),
item-interface (page 712), and semi-tie-column-interface (page 736).
This object is of class Item (characterized by item-interface (page 712)).

3.1.74 LedgerLineSpanner
An auxiliary grob to manage ledger lines of a whole staff.
LedgerLineSpanner objects are created by: Ledger_line_engraver (page 431).
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 #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-extents (_ _ _)> >
    Two skylines, one above and one below this grob.

X-extent (pair of numbers):
    #f
    Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

Y-extent (pair of numbers):
    #f
    Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): grob-interface (page 704), ledger-line-spanner-interface (page 715), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.75 LeftEdge

The left edge of a staff. Useful as an anchor point for other grobs.

LeftEdge objects are created by: Break_align_engraver (page 410).

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 (alist, with symbols as keys):
  '((ambitus extra-space . 1.15)
   (breathing-sign minimum-space . 0.0)
   (cue-end-clef extra-space . 0.8)
   (clef extra-space . 0.8)
   (cue-clef extra-space . 0.8)
   (signum-repetitionis extra-space . 0.0)
   (staff-bar extra-space . 0.0)
   (staff-ellipsis 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.
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): break-aligned-interface (page 687), grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.76 LigatureBracket
A horizontal bracket over a group of notes, usually indicating an ancient ligature if transcribed into modern notation. See also KievanLigature (page 568), MensuralLigature (page 585), and VaticanaLigature (page 667).

LigatureBracket objects are created by: Ligature_bracket_engraver (page 431).

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.

direction (direction):
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 (start . end), where start and end are vertical positions 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).

tuplet-slur (boolean):
    #f
    Draw a slur instead of a bracket for tuplets.

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): grob-interface (page 704), line-interface (page 716), spanner-interface (page 746), and tuplet-bracket-interface (page 760).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.77 LyricExtender
An extender line in lyrics.

LyricExtender objects are created by: Extender_engraver (page 421).

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): grob-interface (page 704), lyric-extender-interface (page 719), lyric-interface (page 720), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.78 LyricHyphen
A hyphen in lyrics. See also VowelTransition (page 673).

LyricHyphen objects are created by: Hyphen_engraver (page 427).

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 (each unit represents half a staff-space).

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 #<procedure ly:grob::vertical-skylines-from-
   stencil (_) #<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): font-interface (page 698),
grob-interface (page 704), lyric-hyphen-interface (page 719), lyric-interface
(page 720), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.79 LyricRepeatCount
A repeat count in lyrics.

LyricRepeatCount objects are created by: Lyric_repeat_count_engraver (page 431).

Standard settings:

break-align-symbols (list):
   ' (staff-bar breathing-sign)
   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):
   #( #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):
\'(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):
\'(-1.0 . 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).

font-series (symbol):
'medium
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.

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.

non-musical (boolean):
#t
True if the grob belongs to a NonMusicalPaperColumn.

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):
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.

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 5575a26ab740 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0 (grob)>
  Text markup. See Section “Formatting text” in Notation Reference.

vertical-skylines (pair of skylines):
  #<unpure-pure-container #<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-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 #<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): break-alignable-interface (page 687),
font-interface (page 698), grob-interface (page 704), item-interface (page 712),
lyric-interface (page 720), lyric-repeat-count-interface (page 720), self-alignment-interface (page 736), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.80 LyricSpace
A space in lyrics.

LyricSpace objects are created by: Hyphen_engraver (page 427).

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):
    #f
    Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.
Y-extent (pair of numbers):

\#f

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): grob-interface (page 704), lyric-hyphen-interface (page 719), lyric-space-interface (page 720), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.81 **LyricText**

A chunk of text in lyrics. See also LyricExtender (page 574), LyricHyphen (page 575), LyricSpace (page 578), and VowelTransition (page 673).

LyricText objects are created by: Lyric_engraver (page 431).

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 5575a26ab720 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363:0
    (grob)>
    Text markup. See Section “Formatting text” in Notation Reference.

vertical-skylines (pair of skylines):
    #<unpure-pure-container #<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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), lyric-syllable-interface
(page 720), rhythmic-grob-interface (page 734), self-alignment-interface (page 736),
and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.82 MeasureCounter
A grob to print a counter for measures.

MeasureCounter objects are created by: Measure_counter_engraver (page 433).

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.
Chapter 3: Backend

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-features (list):
  '("cv47")
  Opentype features.

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.

number-range-separator (markup):
  "-"
  For a measure counter extending over several measures (like with compressed multi-
  measure rests), this is the separator between the two printed numbers.

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):
ly:text-interface::print
The symbol to print.

text (markup):
measure-counter::text
Text markup. See Section “Formatting text” in Notation Reference.

word-space (dimension, in staff space):
0.2
Space to insert between words in texts.

X-offset (number):
centered-spanner-interface::calc-x-offset
The horizontal amount that this object is moved relative to its X-parent.

Y-offset (number):
#<unpure-pure-container #<procedure ly:side-position-interface::y- aligned-side (_ #:optional _)> #<procedure ly:side-position- interface::pure-y-aligned-side (_ _ #:optional _)> >
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): centered-spanner-interface (page 690), font-interface (page 698), grob-interface (page 704), measure-counter-interface (page 720), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.83 MeasureGrouping

A measure grouping or conducting sign.

MeasureGrouping objects are created by: Measure_grouping_ engraver (page 434).

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 #<procedure ly:side-position-interface::y-
  aligned-side (_ #:optional _)> #<procedure ly:side-position-
  interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): grob-interface (page 704),
measure-grouping-interface (page 721), outside-staff-interface (page 730),
side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.84 MeasureSpanner

A horizontal bracket between bar lines. See also HorizontalBracket (page 557).
MeasureSpanner objects are created by: Measure_spanner_engraver (page 434).

Standard settings:

  connect-to-neighbor (pair):
    ly:measure-spanner::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 \text{UP}, \text{CENTER} or \text{DOWN}. Numerical values may also be used: \text{UP}=1, \text{DOWN}=-1, \text{LEFT}=-1, \text{RIGHT}=1, \text{CENTER}=0.

\text{edge-height (pair)}:
\begin{verbatim}
'(0.7 . 0.7)
\end{verbatim}

A pair of numbers specifying the heights of the vertical edges: \textit{(left-height . right-height)}.

\text{outside-staff-priority (number)}:
\begin{verbatim}
750
\end{verbatim}

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 \text{outside-staff-priority} is closer to the staff.

\text{self-alignment-X (number)}:
\begin{verbatim}
0
\end{verbatim}

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.

\text{side-axis (number)}:
\begin{verbatim}
1
\end{verbatim}

If the value is \text{X} (or equivalently 0), the object is placed horizontally next to the other object. If the value is \text{Y} or 1, it is placed vertically.

\text{spacing-pair (pair)}:
\begin{verbatim}
'(staff-bar . staff-bar)
\end{verbatim}

A pair of alignment symbols which set an object’s spacing relative to its left and right \text{BreakAlignments}.

For example, a \text{MultiMeasureRest} will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:
\begin{verbatim}
\override MultiMeasureRest.spacing-pair = #'(staff-bar . staff-bar)
\end{verbatim}

\text{staff-padding (dimension, in staff space)}:
\begin{verbatim}
0.5
\end{verbatim}

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics \text{p} and \text{f}) on their baselines.

\text{stencil (stencil)}:
\begin{verbatim}
ly:measure-spanner::print
\end{verbatim}

The symbol to print.

\text{Y-offset (number)}:
\begin{verbatim}
#<unpure-pure-container #<procedure ly:side-position-interface::yaligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
\end{verbatim}

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): \text{accidental-switch-interface} (page 676), \text{font-interface} (page 698), \text{grob-interface} (page 704), \text{line-interface} (page 716), \text{measure-spanner-interface} (page 721), \text{outside-staff-interface} (page 730), \text{self-alignment-interface} (page 736), \text{side-position-interface} (page 738), \text{spanner-interface} (page 746), and \text{text-interface} (page 755).

This object is of class \text{Spanner} (characterized by \text{spanner-interface} (page 746)).
3.1.85 MelodyItem
An auxiliary grob to help alter the stem directions of middle notes on a staff so that they follow the melody.

MelodyItem objects are created by: Melody_engraver (page 435).

Standard settings:

neutral-direction (direction):
-1
Which direction to take in the center of the staff.

This object supports the following interface(s): grob-interface (page 704), item-interface (page 712), and melody-spanner-interface (page 722).

This object is of class Item (characterized by item-interface (page 712)).

3.1.86 MensuralLigature
A grob to display a ligature as used in mensural notation. See also KievianLigature (page 568), VaticanaLigature (page 667), and LigatureBracket (page 573).

MensuralLigature objects are created by: Mensural_ligature_engraver (page 435).

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): font-interface (page 698), grob-interface (page 704), mensural-ligature-interface (page 722), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.87 MetronomeMark
A metronome mark. This is either a precise tempo indication like ‘quarter note = 80’, or an arbitrary piece of text (like ‘Allegro’), possibly followed by a precise indication in parentheses.

MetronomeMark objects are created by: Metronome_mark_engraver (page 435).

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 it is 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):
  1300
  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 #<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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)>>
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), metronome-mark-interface (page 723), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.88 MultiMeasureRest

A multi-measure rest. See also MultiMeasureRestNumber (page 589), MultiMeasureRestText (page 592), MultiMeasureRestScript (page 590), and Rest (page 611).

MultiMeasureRest objects are created by: Multi_measure_rest_engraver (page 436).

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 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)

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-logs 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 #<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 #<procedure ly:staff-symbol-referrer::callback (_)>>
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698),
      groove-interface (page 704), multi-measure-interface (page 723), multi-measure-rest-
      interface (page 723), outside-staff-interface (page 730), rest-interface (page 733),
      spanner-interface (page 746), and staff-symbol-referencer-interface (page 749).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.89 MultiMeasureRestNumber

A grob to print the length of a MultiMeasureRest (page 587), grob.

MultiMeasureRestNumber objects are created by: Multi_measure_rest_engraver
 (page 436).

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).

font-features (list):
  '("cv47")
  Opentype features.

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.
Chapter 3: Backend

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 #<procedure ly:grob::vertical-skylines-from-
   stencil (_) #<procedure ly:grob::pure-simple-vertical-skylines-from-
   extents (_ _ _) #<procedure ly:grob::pure-y-aligned-side (_ _ #:optional _)
   >
   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 #<procedure ly:grob::stencil-height (_ _) #<procedure
   ly:grob::pure-y-aligned-side (_ _ #:optional _) #<procedure
   ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)
   >
   Extent (size) in the Y direction, measured in staff-space units, relative to object's
   reference point.

Y-offset (number):
   #<unpure-pure-container #<procedure ly:side-position-interface::y-
   aligned-side (_ #:optional _) #<procedure ly:side-position-
   interface::pure-y-aligned-side (_ _ #:optional _) #<procedure
   ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)
   >
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), multi-measure-interface (page 723), multi-measure-
rest-number-interface (page 724), outside-staff-interface (page 730),
self-alignment-interface (page 736), side-position-interface (page 738),
spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.90 MultiMeasureRestScript

An articulation (like a fermata) attached to a MultiMeasureRest (page 587), grob. See also
Script (page 613).

MultiMeasureRestScript objects are created by: Multi_measure_rest_engraver
(page 436).

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-padding (number):
0
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):
40
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.

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:script-interface::print
The symbol to print.

vertical-skylines (pair of skylines):
#<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<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 #<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):

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), multi-measure-interface (page 723), outside-staff-
interface (page 730), script-interface (page 734), self-alignment-interface
(page 736), side-position-interface (page 734), and spanner-interface (page 746).
This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.91 MultiMeasureRestText

A text markup for a MultiMeasureRest (page 587), grob. See also TextScript (page 650).

MultiMeasureRestText objects are created by: Multi_measure_rest_engraver
(page 436).

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.

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.
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 #<procedure ly:grob::vertical-skylines-from-
  stencil (_) #<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 #<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 ly:side-position-interface::y-
  aligned-side (_ #:optional _) #<procedure ly:side-position-
  interface::pure-y-aligned-side (_) _ #:optional _ _ >>
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), multi-measure-interface (page 723), outside-staff-
interface (page 730), self-alignment-interface (page 736), side-position-interface
(page 738), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.92 NonMusicalPaperColumn

An auxiliary grob grouping non-musical items to handle the flexible horizontal space between
non-musical and musical columns. Grobs that have the property non-musical set to #t belong
to this column.

NonMusicalPaperColumn objects are created by: Paper_column_engraver (page 440).

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.

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): axis-group-interface (page 678), font-interface (page 698), grob-interface (page 704), item-interface (page 712), non-musical-paper-column-interface (page 725), paper-column-interface (page 730), separation-item-interface (page 738), and spaceable-grob-interface (page 743).

This object is of class Paper_column (characterized by paper-column-interface (page 730)).
### 3.1.93 NoteCollision

An auxiliary grob to group NoteColumn (page 595), grobs from several voices, mainly to handle note collisions. See also RestCollision (page 612).

NoteCollision objects are created by: Collision_engraver (page 413).

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.

- **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 #<procedure ly:axis-group-interface::height (_)> #<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): axis-group-interface (page 678), grob-interface (page 704), item-interface (page 712), and note-collision-interface (page 726).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.94 NoteColumn

An auxiliary grob to align stacked notes, stems, flags, accidentals, and other items from the same voice. See also NoteCollision (page 595).

NoteColumn objects are created by: Rhythmic_column_engraver (page 445).

Standard settings:

- **axes (list):**
  `'(0 1)`
  List of axis numbers. In the case of alignment grobs, this should contain only one number.
bend-me (boolean):
   '()  
   Decide whether this grob is bent.

horizontal-skylines (pair of skylines):
   ly:separation-item::calc-skylines  
   Two skylines, one to the left and one to the right of this grob.

main-extent (pair of numbers):
   ly:note-column::calc-main-extent  
   The horizontal extent of a NoteColumn grob without taking suspended NoteHead grobs into account (i.e., NoteHeads forced into the unnatural direction of the Stem because of a chromatic clash).

skyline-vertical-padding (number):
   0.15  
   The amount by which the left and right skylines of a column are padded vertically, beyond the Y-extents and extra-spacing-heights of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

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 #<procedure ly:axis-group-interface::height (_)> #<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): axis-group-interface (page 678), bend-interface (page 685), grob-interface (page 704), item-interface (page 712), note-column-interface (page 726), and separation-item-interface (page 738).

This object is of class Item (characterized by item-interface (page 712)).

3.1.95 NoteHead
A note head. See also TabNoteHead (page 648).

NoteHead objects are created by: Completion_heads_engraver (page 414), Drum_notes_engraver (page 419), and Note_heads_engraver (page 438).

Standard settings:

bend-me (boolean):
   '()  
   Decide whether this grob is bent.

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 #<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 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): bend-interface (page 685),
font-interface (page 698), gregorian-ligature-interface (page 702), grob-interface
(page 704), item-interface (page 712), ledgered-interface (page 715), ligature-head-
interface (page 716), mensural-ligature-interface (page 722), note-head-interface
(page 727), rhythmic-grob-interface (page 734), rhythmic-head-interface (page 734),
staff-symbol-referencer-interface (page 749), and vaticana-ligature-interface
(page 762).

This object is of class Item (characterized by item-interface (page 712)).

3.1.96 NoteName
A textual representation of a note name.

NoteName objects are created by: Note_name_engraver (page 438).
Standard settings:

stencil (stencil):
  ly:text-interface::print
  The symbol to print.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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): accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), item-interface (page 712), note-name-interface (page 728), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.97 NoteSpacing

An auxiliary grob to handle (horizontal) spacing of notes. See also GraceSpacing (page 553), StaffSpacing (page 632), and SpacingSpanner (page 626).

NoteSpacing objects are created by: Note_spacing_engraver (page 439).

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): grob-interface (page 704), item-interface (page 712), note-spacing-interface (page 728), and spacing-interface (page 744).

This object is of class Item (characterized by item-interface (page 712)).
3.1.98 OttavaBracket

An ottava bracket.

OttavaBracket objects are created by: Ottava_spanner_engraver (page 439).

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.

- **edge-height** (pair):
  - `(0.8)
  - A pair of numbers specifying the heights of the vertical edges: `(left-height, right-height)`.

- **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.

- **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 #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-extents (_ _ _)> >

Two skylines, one above and one below this grob.

Y-offset (number):
#<unpure-pure-container #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698), grob-interface (page 704), horizontal-bracket-interface (page 709), line-interface (page 716), ottava-bracket-interface (page 729), outside-staff-interface (page 730), side-position-interface (page 738), spanner-interface (page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.99 PaperColumn
An auxiliary grob grouping musical items to handle the flexible horizontal space between musical and non-musical columns. See also NonMusicalPaperColumn (page 593).

PaperColumn objects are created by: Paper_column_engraver (page 440).

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.

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.

\texttt{skyline-vertical-padding (number): 0.08}

The amount by which the left and right skylines of a column are padded vertically, beyond the Y-extents and extra-spacing-heights of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

\texttt{X-extent (pair of numbers):}

\texttt{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): \texttt{axis-group-interface} (page 678), \texttt{font-interface} (page 698), \texttt{grob-interface} (page 704), \texttt{item-interface} (page 712), \texttt{musical-paper-column-interface} (page 724), \texttt{paper-column-interface} (page 730), \texttt{separation-item-interface} (page 738), and \texttt{spaceable-grob-interface} (page 743).

This object is of class \texttt{Paper\_column} (characterized by \texttt{paper-column-interface} (page 730)).

### 3.1.100 Parentheses

A grob to create parentheses around other grobs.

Parentheses objects are created by: \texttt{Parenthesis\_engraver} (page 440).

Standard settings:

\texttt{break-visibility (vector):}

\texttt{#<procedure 5575a26ab700 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:2973:0 (grob)>}

A vector of 3 booleans, \texttt{(#end-of-line unbroken begin-of-line)}. \texttt{#t} means visible, \texttt{#f} means killed.

\texttt{font-size (number):}

\texttt{-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 \texttt{fontSize} is set, its value is added to this before the glyph is printed. Fractional values are allowed.

\texttt{padding (dimension, in staff space):}

\texttt{0.2}

Add this much extra space between objects that are next to each other.

\texttt{stencil (stencil):}

\texttt{parentheses-interface::print}

The symbol to print.

\texttt{stencils (list):}

\texttt{parentheses-interface::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-interface::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): font-interface (page 698), grob-interface (page 704), parentheses-interface (page 731), and sticky-grob-interface (page 752).

This object can be of either of the following classes: Item (characterized by item-interface) or Spanner (characterized by spanner-interface). It supports the following interfaces conditionally depending on the class: item-interface (page 712), and spanner-interface (page 746).

3.1.101 PercentRepeat
A percent symbol for repeating a bar. See also PercentRepeatCounter (page 603), DoublePercentRepeat (page 532), DoubleRepeatSlash (page 535), and RepeatSlash (page 609).

PercentRepeat objects are created by: Percent_repeat_engraver (page 441).

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).

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.

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 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)
springs-and-rods (boolean):
   ly:multi-measure-rest::set-spacing-rods
   Dummy variable for triggering spacing routines.

stencil (stencil):
   ly:percent-repeat-interface::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).

X-offset (number):
   centered-spanner-interface::calc-x-offset
   The horizontal amount that this object is moved relative to its X-parent.

   This object supports the following interface(s): centered-spanner-interface (page 690),
   font-interface (page 698), grob-interface (page 704), multi-measure-rest-interface
   (page 723), percent-repeat-interface (page 731), and spanner-interface (page 746).

   This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.102 PercentRepeatCounter
A grob to print a counter for PercentRepeat (page 602), grobs.

   PercentRepeatCounter objects are created by: Percent_repeat_engraver (page 441).

   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-features (list):
      '("cv47")
      Opentype features.

   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.

display-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):
   ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-
   interface::pure-y-aligned-side (_ _ #:optional _)> >
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), outside-staff-interface (page 730), self-alignment-
interface (page 736), side-position-interface (page 738), spanner-interface
(page 746), and text-interface (page 755).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.103 PhrasingSlur

A phrasing slur, indicating a ‘musical sentence’. See also Slur (page 621).

PhrasingSlur objects are created by: Phrasing_slur_engraver (page 441).

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 (alist, with symbols as keys):
'((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))

A list 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 #<procedure ly:grob::vertical-skylines-from-
      stencil (_)> #<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 #<procedure ly:slur::height (_)> #<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): bezier-curve-interface (page 687),
grob-interface (page 704), outside-staff-interface (page 730), slur-interface
(page 741), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.104 PianoPedalBracket
A piano pedal bracket. It can also be part of SostenutoPedal (page 623), SustainPedal
(page 641), or UnaCordaPedal (page 665), grobs if they are printed in a bracketed style.

PianoPedalBracket objects are created by: Piano_pedal_engraver (page 442).

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 #<procedure ly:grob::vertical-skylines-from-
stencil (_)> #<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): grob-interface (page 704),
line-interface (page 716), piano-pedal-bracket-interface (page 732), piano-pedal-
interface (page 732), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.105 RehearsalMark

A rehearsal mark.

RehearsalMark objects are created by: Mark_engraver (page 432).

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):**

break-alignable-interface::self-alignment-opposite-of-anchor

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.
The symbol to print.

Two skylines, one above and one below this grob.

The horizontal amount that this object is moved relative to its X-parent.

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface (page 676), break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), mark-interface (page 720), outside-staff-interface (page 730), rehearsal-mark-interface (page 733), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.106 RepeatSlash

A symbol consisting of one or more slashes for repeating patterns shorter than a single measure, and which contain identical durations. See also PercentRepeat (page 602), DoublePercentRepeat (page 532), and DoubleRepeatSlash (page 535).

RepeatSlash objects are created by: Slash_repeat_engraver (page 446).

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-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 #<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): grob-interface (page 704), item-interface (page 712), percent-repeat-interface (page 731), and rhythmic-grob-interface (page 734).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.107 RepeatTie

A repeat tie (i.e., a tie from nothing to a note). See also RepeatTieColumn (page 611), LaissezVibrerTie (page 569), and Tie (page 653).

RepeatTie objects are created by: Repeat_tie_engraver (page 444).

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` (alist, with symbols as keys):
  ```
  '((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.
Chapter 3: Backend

3.1.108 RepeatTieColumn
An auxiliary grob to determine direction and shape of stacked RepeatTie (page 610), grobs.

RepeatTieColumn objects are created by: Repeat_tie_engraver (page 444).

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):
  #f
  Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

- Y-extent (pair of numbers):
  #f
  Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): grob-interface (page 704), item-interface (page 712), and semi-tie-column-interface (page 736).

This object is of class Item (characterized by item-interface (page 712)).

3.1.109 Rest
An ordinary rest. See also MultiMeasureRest (page 587).

Rest objects are created by: Completion_rest_engraver (page 414), and Rest_engraver (page 444).

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 #<procedure ly:grob::vertical-skylines-from-
    stencil (_) > #<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 #<procedure ly:rest::height (_) > #<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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), rest-interface (page 733),
rhythmic-grob-interface (page 734), rhythmic-head-interface (page 734), and
staff-symbol-referencer-interface (page 749).
This object is of class Item (characterized by item-interface (page 712)).

3.1.110 RestCollision
An auxiliary grob to handle rest collisions of different voices. See also NoteCollision (page 595).

RestCollision objects are created by: Rest_collision_ engraver (page 444).

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): grob-interface (page 704), item-interface (page 712), and rest-collision-interface (page 733).

This object is of class Item (characterized by item-interface (page 712)).

3.1.111 Script

An articulation (staccato, accent, etc.). See also ScriptColumn (page 614), ScriptRow (page 614), and MultiMeasureRestScript (page 590).

Script objects are created by: Drum_notes_engraver (page 419), New_fingering_engraver (page 437), and Script_engraver (page 445).

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 \texttt{p} and \texttt{f}) on their baselines.
stencil (stencil):
    ly:script-interface::print
    The symbol to print.
vertical-skylines (pair of skylines):
    #<unpure-pure-container #<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 #<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 ly:side-position-interface::y-
       aligned-side (_ #:optional _)> #<procedure ly:side-position-
       interface::pure-y-aligned-side (_ _ #:optional _)> >
    The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), outside-staff-interface
(page 730), script-interface (page 734), self-alignment-interface (page 736), and
side-position-interface (page 738).
This object is of class Item (characterized by item-interface (page 712)).

3.1.112 ScriptColumn
An auxiliary grob to (vertically) align stacked Script (page 613), grobs.
ScriptColumn objects are created by: Non_musical_script_column_engraver (page 437),
and Script_column_engraver (page 445).
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): grob-interface (page 704),
item-interface (page 712), and script-column-interface (page 734).
This object is of class Item (characterized by item-interface (page 712)).

3.1.113 ScriptRow
An auxiliary grob to horizontally align stacked Script (page 613), grobs.
ScriptRow objects are created by: Script_row_engraver (page 445).
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): grob-interface (page 704),
item-interface (page 712), and script-column-interface (page 734).
This object is of class Item (characterized by item-interface (page 712)).
SectionLabel

A section label, for example ‘Coda’.

SectionLabel objects are created by: Mark_engraver (page 432).

Standard settings:

after-line-breaking (boolean):
- lyside-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):
- ‘(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 #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):
- 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.

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):
  1450
  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.

stencil (stencil):
  ly:text-interface::print
  The symbol to print.

vertical-skylines (pair of skylines):
  #<unpure-pure-container #<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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)>>
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), outside-staff-interface (page 730), section-label-interface (page 735), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.115 SegnoMark

A segno mark (created with \repeat segno, not with \segno).

SegnoMark objects are created by: Mark_engraver (page 432).

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):
  1400
  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):
break-alignable-interface::self-alignment-opposite-of-anchor
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 #<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 #<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 ly:side-position-interface::y-aligned-side (_ #:optional _) #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): break-alignable-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), mark-interface (page 720), outside-staff-interface (page 730), segno-mark-interface (page 736), self-alignment-interface (page 736), side-position-interface (page 738), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.116 SignumRepetitionis
SignumRepetitionis objects are created by: Signum_repetitionis_engraver (page 446).

Standard settings:

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):
'signum-repetitionis
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::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):
  #<procedure 5575a26ab6e0 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1418:0 (grob)> 
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):
    #f
    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 (alist, with symbols as keys):
    '(((ambitus extra-space . 1.0)
       (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 extra-space . 0.5)
       (next-note semi-fixed-space . 0.9)
       (signum-repetitionis extra-space . 0.5)
       (staff-bar extra-space . 0.5)
       (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.

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 #<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): break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), pure-from-neighbor-interface (page 733), and signum-repetition-is-interface (page 740).

This object is of class Item (characterized by item-interface (page 712)).

3.1.117 Slur
A slur. See also PhrasingSlur (page 604).

Slur objects are created by: Slur_engraver (page 446).

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 (alist, with symbols as keys):
'(('region-size . 4)
A list 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 #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<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 #<procedure ly:slur::height (_)> #<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): bezier-curve-interface (page 687), grob-interface (page 704), outside-staff-interface (page 730), slur-interface (page 741), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.118 SostenutoPedal
A sostenuto pedal mark. See also SostenutoPedallineSpanner (page 624), PianoPedalBracket (page 606), SustainPedal (page 641), and UnaCordaPedal (page 665).

SostenutoPedal objects are created by: Piano_pedal_engraver (page 442).

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):
  #f
  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 #<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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), piano-pedal-script-interface
(page 732), self-alignment-interface (page 736), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.119 SostenutoPedalLineSpanner

An auxiliary grob providing a baseline to align consecutive SostenutoPedal (page 623), grobs
vertically.

SostenutoPedalLineSpanner objects are created by: Piano_pedal_align_engraver
(page 442).

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 #<procedure ly:grob::vertical-skylines-from-element-stencils (_)> #<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 #<procedure ly:axis-group-interface::height (_)> #<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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
The vertical amount that this object is moved relative to its Y-parent.
This object supports the following interface(s): axis-group-interface (page 678),
grob-interface (page 704), outside-staff-interface (page 730), piano-pedal-interface (page 732), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.120 SpacingSpanner

An auxiliary grob to set all horizontal spacing constraints across a score. There is normally one such grob for the whole score, but there can be several if \newSpacingSection is used. See also GraceSpacing (page 553), NoteSpacing (page 598), and StaffSpacing (page 632).

SpacingSpanner objects are created by: Spacing_engraver (page 447).

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): grob-interface (page 704),
 spacing-options-interface (page 744), spacing-spanner-interface (page 744), and
 spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.121 SpanBar

A span bar, i.e., the parts of a multi-staff bar line that are outside of staves. See also SpanBarStub (page 628).

SpanBar objects are created by: Span_bar_engraver (page 448).

Standard settings:

allow-span-bar (boolean):
  #t
  If false, no inter-staff bar line will be created below this bar line.
Chapter 3: Backend

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.

before-line-breaking (boolean):

Dummy property, used to trigger a callback function.

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-symbol (symbol):

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in Internals Reference.

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.

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.

non-musical (boolean):

True if the grob belongs to a NonMusicalPaperColumn.

stencil (stencil):

The symbol to print.

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): bar-line-interface (page 681), break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), and span-bar-interface (page 745).

This object is of class Item (characterized by item-interface (page 712)).
3.1.122 SpanBarStub
An auxiliary grob, acting like a fake SpanBar (page 626), grob in contexts such as Lyrics (page 199), that are crossed by a span bar, to keep span bars taking horizontal space.

SpanBarStub objects are created by: Span_bar_stub_engraver (page 448).

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 5575a25ef000 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1407:0 (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): grob-interface (page 704), item-interface (page 712), and pure-from-neighbor-interface (page 733).

This object is of class Item (characterized by item-interface (page 712)).

3.1.123 StaffEllipsis
A visual marker (usually three consecutive dots) to indicate that typesetting of music is skipped.

StaffEllipsis objects are created by: Skip_typesetting_engraver (page 446).

Standard settings:

break-align-symbol (symbol):
    'staff-ellipsis
    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.

layer (integer):
    1
    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.

space-alist (alist, with symbols as keys):
    '(((ambitus extra-space . 1.0)
        (breathing-sign extra-space . 1.0)
        (custos extra-space . 1.0)
        (key-signature extra-space . 1.0)
        (time-signature extra-space . 1.0)
        (signum-repetitionis extra-space . 1.0)
        (staff-bar extra-space . 1.0)
        (clef extra-space . 1.0)
        (cue-clef extra-space . 1.0)
        (cue-end-clef extra-space . 1.0)
        (first-note extra-space . 1.0)
        (right-edge fixed-space . 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):
staff-ellipsis::print
The symbol to print.

text (markup):
'(#<procedure line-markup (layout props args>>
  (#<procedure null-markup (layout props)>>
  (#<procedure musicglyph-markup (layout props glyph-name)> "dots.dot")
  (#<procedure musicglyph-markup (layout props glyph-name)> "dots.dot")
  (#<procedure musicglyph-markup (layout props glyph-name)> "dots.dot")
  (#<procedure null-markup (layout props)>>>)

Text markup. See Section “Formatting text” in Notation Reference.

whiteout (boolean-or-number):
#t
If a number or true, the grob is printed over a white background to white-out underly-ing 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.

Y-extent (pair of numbers):
staff-ellipsis::calc-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): break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.124 StaffGrouper
An auxiliary grob to manage vertical spacing of staff groups. See also VerticalAlignment (page 668), and VerticalAxisGroup (page 668).

StaffGrouper objects are created by: Vertical_align_engraver (page 456).

Standard settings:

staff-staff-spacing (alist, with symbols as keys):
'((basic-distance . 9)
 (minimum-distance . 7)
 (padding . 1)
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** (alist, with symbols as keys):

`'((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): grob-interface (page 704), spanner-interface (page 746), and staff-grouper-interface (page 747).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.125 **StaffHighlight**

A colored span to highlight a music passage.

**StaffHighlight** objects are created by: **Staff_highlight_engraver** (page 449).

Standard settings:

- **bound-prefatory-paddings** (pair of numbers):
  
  `'(0.5 0.5)`

  For a highlight, the amount of padding to insert at a bound from a prefatory item that is not a bar line.

- **color** (color):

  ```lily
  #<procedure 5575a26ab6c0 at /build/out/share/lilypond/current/scm/lily/output-lib.scm:1363> (grob)
  ```

  The color of this grob.

- **layer** (integer):

  `-1`
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.

shorten-pair (pair of numbers):

'(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):
staff-highlight::*print
The symbol to print.

X-extent (pair of numbers):
staff-highlight::*width
Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

Y-extent (pair of numbers):
staff-highlight::*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): grob-interface (page 704), spanner-interface (page 746), and staff-highlight-interface (page 747).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.126 StaffSpacing

An auxiliary grob to handle spacing within a staff. See also NoteSpacing (page 598), GraceSpacing (page 553), and SpacingSpanner (page 626).

StaffSpacing objects are created by: Separating_line_group_engraver (page 445).

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): grob-interface (page 704), item-interface (page 712), spacing-interface (page 744), and staff-spacing-interface (page 748).

This object is of class Item (characterized by item-interface (page 712)).

3.1.127 StaffSymbol

A staff symbol, usually five horizontal lines.

StaffSymbol objects are created by: Staff_symbol_engraver (page 449), and Tab_staff_symbol_engraver (page 451).
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.

widened-extent (pair of numbers):
  staff-symbol::calc-widened-extent
The vertical extent that a bar line on a certain staff symbol should have. If the staff symbol is small (e.g., has just one line, as in a RhythmicStaff, this is wider than the staff symbol’s Y extent.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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): grob-interface (page 704), spanner-interface (page 746), and staff-symbol-interface (page 748).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.128 StanzaNumber

A stanza number (or markup) for lyrics.

StanzaNumber objects are created by: Stanza_number_ engraver (page 450).

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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), side-position-interface
(page 738), stanza-number-interface (page 749), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.129 Stem
A stem. See also StemStub (page 636).

  Stem objects are created by: Span_stem_engraver (page 448), and Stem_engraver
  (page 450).

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.
A list 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: \( \text{UP}=1, \text{DOWN}=-1, \text{LEFT}=-1, \text{RIGHT}=1, \text{CENTER}=0. \)

**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., \( 0 = \text{whole note}, 1 = \text{half note}, \text{etc.} \)

**length** (dimension, in staff space):

- `#<unpure-pure-container #<procedure ly:stem::calc-length (_)> #<procedure ly:stem::pure-calc-length (_, _)> >`

  User override for the stem length of unbeamed stems (each unit represents half a staff-space).

**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 #<procedure ly:stem::calc-stem-begin-position (_)> #<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):
   ly:stem::height (_)
   ly:stem::pure-height (_ _ _)
   Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

Y-offset (number):
   ly:stem::pure-height (_ _ _)
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): grob-interface (page 704), item-interface (page 712), and stem-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.130 StemStub
An auxiliary grob that prevents cross-staff Stem (page 634), grobs from colliding with articulations.

StemStub objects are created by: Stem_engraver (page 450).

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):
   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): grob-interface (page 704), and item-interface (page 712).

This object is of class Item (characterized by item-interface (page 712)).

3.1.131 StemTremolo

A stem tremolo.

StemTremolo objects are created by: Stem_engraver (page 450).

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):

\[
\text{Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.}
\]

Y-offset (number):

\[
\text{The vertical amount that this object is moved relative to its Y-parent.}
\]

This object supports the following interface(s): grob-interface (page 704), item-interface (page 712), self-alignment-interface (page 736), and stem-tremolo-interface (page 751).

This object is of class Item (characterized by item-interface (page 712)).

3.1.132 StringNumber

A markup (by default a digit in a circle) to name a string.

StringNumber objects are created by: New_fingering_engraver (page 437).

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.

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-features (list):

- "cv47"
  Opentype features.

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 arabic, roman-ij-lower, roman-ij-upper, roman-lower, and roman-upper.
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 #<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): font-interface (page 698), grob-interface (page 704), item-interface (page 712), number-interface (page 728), outside-staff-interface (page 730), self-alignment-interface (page 736), side-position-interface (page 738), string-number-interface (page 752), text-interface (page 755), and text-script-interface (page 756).

This object is of class Item (characterized by item-interface (page 712)).
3.1.133 **StrokeFinger**

A markup (usually a lowercase letter) to indicate right-hand fingering. See also Fingering (page 545).

*StrokeFinger* objects are created by: `New_fingering_engraver` (page 437).

Standard settings:

- **add-stem-support** (boolean):
  - `only-if-beamed`
    
    If set, the *Stem* object is included in this script’s support.

- **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 #<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): font-interface (page 698),
   grob-interface (page 704), item-interface (page 712), outside-staff-interface
   (page 730), self-alignment-interface (page 736), side-position-interface
   (page 738), stroke-finger-interface (page 752), text-interface (page 755), and
   text-script-interface (page 756).

This object is of class Item (characterized by item-interface (page 712)).

3.1.134 SustainPedal

A sustain pedal mark. See also SustainPedalLineSpanner (page 642), PianoPedalBracket
   (page 606), SostenutoPedal (page 623), and UnaCordaPedal (page 665).

SustainPedal objects are created by: Piano_pedal_engraver (page 442).

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):
   #f
   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 #<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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), piano-pedal-interface
(page 732), piano-pedal-script-interface (page 732), self-alignment-interface
(page 736), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.135 SustainPedalLineSpanner
An auxiliary grob providing a baseline to align consecutive SustainPedal (page 641), grobs vertically.

SustainPedallineSpanner objects are created by: Piano_pedal_align_engraver
(page 442).

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 #<procedure ly:grob::vertical-skylines-from-element-stencils (_)> #<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 #<procedure ly:axis-group-interface::height (_)> #<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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
   The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): axis-group-interface (page 678),
grob-interface (page 704), outside-staff-interface (page 730), piano-pedal-interface (page 732), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.136 System
The top-level grob of a score. All other grobs are descendants of it.

System objects are created internally by the Score_engraver translator group.

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.

**show-vertical-skylines** (boolean):
  
  grob::show-skylines-if-debug-skylines-set
  
  If true, print this grob’s vertical skylines. This is meant for debugging purposes.

**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 #<procedure ly:system::height (_) #<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): **axis-group-interface** (page 678), **grob-interface** (page 704), **outside-staff-axis-group-interface** (page 729), **spanner-interface** (page 746), and **system-interface** (page 753).

This object is of class **System** (characterized by **system-interface** (page 753)).

### 3.1.137 SystemStartBar

A bar line as a system start delimiter.

SystemStartBar objects are created by: **System_start_delimiter_ engraver** (page 450).

**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): grob-interface (page 704), side-position-interface (page 738), spanner-interface (page 746), and system-start-delimiter-interface (page 753).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.138 SystemStartBrace
A brace as a system start delimiter.
SystemStartBrace objects are created by: System_start_delimiter_engraver (page 450).

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): font-interface (page 698),
  grob-interface (page 704), side-position-interface (page 738), spanner-interface
  (page 746), and system-start-delimiter-interface (page 753).

  This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.139 SystemStartBracket
A bracket as a system start delimiter.

  SystemStartBracket objects are created by: System_start_delimiter_engraver
  (page 450).

  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): font-interface (page 698), grob-interface (page 704), side-position-interface (page 738), spanner-interface (page 746), and system-start-delimiter-interface (page 753).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.140 SystemStartSquare
A rectangle-like bracket as a start delimiter.

SystemStartSquare objects are created by: System_start_delimiter_engraver (page 450).

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): font-interface (page 698),
grob-interface (page 704), side-position-interface (page 738), spanner-interface
(page 746), and system-start-delimiter-interface (page 753).

This object is of class Spanner (characterized by spanner-interface (page 746)).

### 3.1.141 TabNoteHead

A ‘note head’ (usually a digit) in a tablature. See also NoteHead (page 596).

TabNoteHead objects are created by: Tab_note_heads_engraver (page 451).

Standard settings:

- **bend-me** (boolean):
  - ()
    Decide whether this grob is bent.
- **details** (alist, with symbols as keys):
  - '((cautionary-properties
      (angularity . 0.4)
      (half-thickness . 0.075)
      (padding . 0)
      (procedure
        .
        #<procedure parenthesize-stencil (stencil half-thickness width angularity padding)
          (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 angularity padding)
              (width . 0.25))
            (repeat-tied-properties
              (note-head-visible . #t)
              (parenthesize . #t))
            (tied-properties (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):`

`ly:note-head::calc-tab-stem-attachment`

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 #<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 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): `bend-interface` (page 685), `font-interface` (page 698), `grob-interface` (page 704), `item-interface` (page 712), `note-head-interface` (page 727), `rhythmic-grob-interface` (page 734), `rhythmic-head-interface` (page 734), `staff-symbol-referencer-interface` (page 749), `tab-note-head-interface` (page 754), and `text-interface` (page 755).

This object is of class Item (characterized by `item-interface` (page 712)).
3.1.142 TextScript

A markup attached to a grob like a note head. See also MultiMeasureRestText (page 592).

TextScript objects are created by: Text_engraver (page 452).

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 it is 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):
  - #f
    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):
  #f
  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 #<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 #<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 ly:side-position-interface::y-
  aligned-side (_) #:optional _)> #<procedure ly:side-position-
  interface::pure-y-aligned-side (_) _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): accidental-switch-interface
(page 676), font-interface (page 698), grob-interface (page 704), instrument-specific-
markup-interface (page 710), item-interface (page 712), outside-staff-interface
(page 730), self-alignment-interface (page 736), side-position-interface (page 738),
text-interface (page 755), and text-script-interface (page 756).

This object is of class Item (characterized by item-interface (page 712)).
3.1.143 TextSpanner

Text like 'rit', usually followed by a (dashed) line. See also DynamicTextSpanner (page 541).

TextSpanner objects are created by: Text_spanner_engraver (page 452).

Standard settings:

bound-details (alist, with symbols as keys):
   '('
   (left (padding . 0.25) (attach-dir . -1))
   (left-broken (attach-dir . 1))
   (right (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 (alist, with symbols as keys):
   ly:horizontal-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 (alist, with symbols as keys):
   ly:horizontal-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 #<procedure ly:side-position-interface::y-aligned-side (_ #:optional _)> #<procedure ly:side-position-interface::pure-y-aligned-side (_ _ #:optional _)> >
  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): font-interface (page 698), grob-interface (page 704), horizontal-line-spanner-interface (page 710), line-interface (page 716), line-spanner-interface (page 717), outside-staff-interface (page 730), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.144 Tie

A tie. See also TieColumn (page 655), LaissezVibrerTie (page 569), and RepeatTie (page 610).

Tie objects are created by: Completion_heads_ engraver (page 414), and Tie_ engraver (page 452).

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 (alist, with symbols as keys):
  '((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)
A list 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.

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 #<procedure ly:grob::vertical-skylines-from-stencil (_) #<procedure ly:grob::pure-simple-vertical-skylines-from-extents (_ _) #<procedure ly:grob::vertical-skylines-from-stencil (_) #<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): bezier-curve-interface (page 687),
grob-interface (page 704), spanner-interface (page 746), and tie-interface (page 757).
This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.145 TieColumn

An auxiliary grob to determine direction and shape of stacked Tie (page 653), grobs.
TieColumn objects are created by: Completion_heads_engraver (page 414), and Tie_engraver (page 452).
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):
    #f
    Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

Y-extent (pair of numbers):
    #f
    Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): grob-interface (page 704),
spanner-interface (page 746), and tie-column-interface (page 756).
This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.146 TimeSignature

A time signature.
TimeSignature objects are created by: Time_signature_engraver (page 453).
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 (alist, with symbols as keys):
  '((ambitus extra-space . 1.0)
   (cue-clef extra-space . 1.5)
   (first-note fixed-space . 2.0)
   (right-edge extra-space . 0.5)
   (signum-repetitionis extra-space . 1.0)
   (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 \textit{spacing-style} are:

\begin{itemize}
\item \texttt{extra-space}
  \begin{itemize}
  \item Put this much space between the two grobs. The space is stretchable when paired with \texttt{first-note} or \texttt{next-note}; otherwise it is fixed.
  \end{itemize}
\item \texttt{minimum-space}
  \begin{itemize}
  \item 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 \texttt{first-note} or \texttt{next-note}; otherwise it is fixed. Not compatible with \texttt{right-edge}.
  \end{itemize}
\item \texttt{fixed-space}
  \begin{itemize}
  \item Only compatible with \texttt{first-note} and \texttt{next-note}. Put this much fixed space between the grob and the note.
  \end{itemize}
\item \texttt{minimum-fixed-space}
  \begin{itemize}
  \item Only compatible with \texttt{first-note} and \texttt{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.
  \end{itemize}
\item \texttt{semi-fixed-space}
  \begin{itemize}
  \item Only compatible with \texttt{first-note} and \texttt{next-note}. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.
  \end{itemize}
\end{itemize}

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

\texttt{stencil (stencil)}: \\
\begin{verbatim}
  \texttt{ly:time-signature::print}
\end{verbatim}

The symbol to print.

\texttt{style (symbol)}:
\begin{verbatim}
  '\C'
\end{verbatim}

This setting determines in what style a grob is typeset. Valid choices depend on the \texttt{stencil} callback reading this property.

\texttt{Y-extent (pair of numbers)}:
\begin{verbatim}
  #<unpure-pure-container #<procedure ly:grob::stencil-height (_)> >
\end{verbatim}

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): break-aligned-interface (page 687), font-interface (page 698), grob-interface (page 704), item-interface (page 712), pure-from-neighbor-interface (page 733), and time-signature-interface (page 759).

This object is of class \texttt{Item} (characterized by \texttt{item-interface} (page 712)).

\subsection*{3.1.147 \texttt{TrillPitchAccidental}}

The accidental of a pitched trill. See also \texttt{TrillPitchGroup} (page 658).

\texttt{TrillPitchAccidental} objects are created by: \texttt{Pitched_trill_ engraver} (page 443).

Standard settings:

\begin{verbatim}
  \texttt{direction (direction)}: \\
  \begin{verbatim}
    -1
  \end{verbatim}
\end{verbatim}

If \texttt{side-axis} is 0 (or X), then this property determines whether the object is placed \texttt{LEFT}, \texttt{CENTER} or \texttt{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.

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 #<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): accidental-interface (page 675), accidental-switch-interface (page 676), font-interface (page 698), grob-interface (page 704), inline-accidental-interface (page 710), item-interface (page 712), side-position-interface (page 738), and trill-pitch-accidental-interface (page 760).

This object is of class Item (characterized by item-interface (page 712)).

3.1.148 TrillPitchGroup
An auxiliary grob to construct a pitched trill, aligning TrillPitchAccidental (page 657), TrillPitchParentheses (page 660), and TrillPitchHead (page 659), horizontally. See also TrillSpanner (page 661).

TrillPitchGroup objects are created by: Pitched_trill_ engraver (page 443).

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.

**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.

**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.

**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 #<procedure ly:axis-group-interface::height (_)> #<procedure trill-pitch-group::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): `axis-group-interface` (page 678), `grob-interface` (page 704), `item-interface` (page 712), and `side-position-interface` (page 738).

This object is of class Item (characterized by `item-interface` (page 712)).

### 3.1.149 TrillPitchHead

The note head of a pitched trill. See also TrillPitchGroup (page 658).

TrillPitchHead objects are created by: `Pitched_trill_engraver` (page 443).

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.

parenthesis-friends (list):
  '(accidental-grob)
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:note-head::print
The symbol to print.

Y-extent (pair of numbers):
  #<unpure-pure-container #<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 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): font-interface (page 698), grob-interface (page 704), item-interface (page 712), ledgered-interface (page 715), pitched-trill-interface (page 732), rhythmic-head-interface (page 734), and staff-symbol-referencer-interface (page 749).

This object is of class Item (characterized by item-interface (page 712)).

3.1.150 TrillPitchParentheses

The parentheses of a pitched trill. See also TrillPitchGroup (page 658).

TrillPitchParentheses objects are created by: Pitched_trill_ engraver (page 443).

Standard settings:

  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):
    0.3
    Add this much extra space between objects that are next to each other.

  stencil (stencil):
    parentheses-interface::print
    The symbol to print.

  stencils (list):
    parentheses-interface::calc-parenthesis-stencils
    Multiple stencils, used as intermediate value.
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): font-interface (page 698), grob-interface (page 704), item-interface (page 712), parentheses-interface (page 731), and pitched-trill-interface (page 732).

This object is of class Item (characterized by item-interface (page 712)).

### 3.1.151 TrillSpanner

A continued trill with a wiggly line (created with \startTrillSpan, not with \trill). See also TrillPitchGroup (page 658).

TrillSpanner objects are created by: Trill_spanner_engraver (page 455).

Standard settings:

after-line-breaking (boolean):

ly:spanner::kill-zero-spanned-time

Dummy property, used to trigger callback for after-line-breaking.

bound-details (alist, with symbols as keys):

'(left (text #<procedure with-true-dimension-markup (layout props axis arg)> 0
    (#<procedure musicglyph-markup (layout props glyph-name)>
    "scripts.trill")
    (stencil-offset 0 . -1)
    (attach-dir . 0))
 (left-broken (end-on-note . #t))
(right (adjust-on-neighbor . #t)
    (attach-dir . -1)
    (end-on-accidental . #t)))

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 (alist, with symbols as keys):

ly:horizontal-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 (alist, with symbols as keys):
  ly:horizontal-line-spanner::calc-right-bound-info
  An alist of properties for determining attachments of spanners to edges.

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.

to-barline (boolean):
  #t
  If true, the spanner will stop at the bar line just before it would otherwise stop.

This object supports the following interface(s): font-interface (page 698),
grob-interface (page 704), horizontal-line-spanner-interface (page 710),
line-interface (page 716), line-spanner-interface (page 717), outside-staff-interface (page 730), side-position-interface (page 738), spanner-interface (page 746), and trill-spanner-interface (page 760).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.152 TupletBracket
A tuplet bracket. See also TupletNumber (page 664).

TupletBracket objects are created by: Tuplet_engraver (page 455).

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.

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 (start . end), where start and end are vertical positions
   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).

tuplet-slur (boolean):
   #f
   Draw a slur instead of a bracket for tuplets.

vertical-skylines (pair of skylines):
   #<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-stencil (_)> #<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): grob-interface (page 704),
line-interface (page 716), outside-staff-interface (page 730), spanner-interface
(page 746), and tuplet-bracket-interface (page 760).
This object is of class Spanner (characterized by spanner-interface (page 746)).
3.1.153 **TupletNumber**

A tuplet number. See also TupletBracket (page 662).

TupletNumber objects are created by: Tuplet_engraver (page 455).

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):

```csharp
  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):

```csharp
  ly:tuplet-number::print

The symbol to print.
```

`text` (markup):

```csharp
  tuplet-number::calc-denominator-text

Text markup. See Section “Formatting text” in Notation Reference.
```

`X-offset` (number):

```csharp
  ly:tuplet-number::calc-x-offset

The horizontal amount that this object is moved relative to its X-parent.
```

`Y-offset` (number):

```csharp
  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): font-interface (page 698),

- grob-interface (page 704),
- outside-staff-interface (page 730),
- spanner-interface (page 746),
- text-interface (page 755),
- and tuplet-number-interface (page 761).

This object is of class Spanner (characterized by spanner-interface (page 746)).
3.1.154 UnaCordaPedal

An una corda pedal mark. See also UnaCordaPedalLineSpanner (page 666), SostenutoPedal (page 623), SustainPedal (page 641), and PianoPedalBracket (page 606).

UnaCordaPedal objects are created by: Piano_pedal_engraver (page 442).

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):
   #f
   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 #<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 #<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): font-interface (page 698),
grob-interface (page 704), item-interface (page 712), piano-pedal-script-interface
(page 732), self-alignment-interface (page 736), and text-interface (page 755).

This object is of class Item (characterized by item-interface (page 712)).

3.1.155 UnaCordaPedallLineSpanner
An auxiliary grob providing a baseline to align consecutive UnaCordaPedal (page 665), grobs
vertically.

UnaCordaPedallLineSpanner objects are created by: Piano_pedal_align_engraver
(page 442).

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 #<procedure ly:grob::vertical-skylines-from-
  element-stencils (_)> #<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 #<procedure ly:axis-group-interface::height
  (_)> #<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 #<procedure ly:side-position-interface::y-
  aligned-side (_ #:optional _)> #<procedure ly:side-position-
  interface::pure-y-aligned-side (_ _ #:optional _)> >

  The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): axis-group-interface (page 678),
grob-interface (page 704), outside-staff-interface (page 730), piano-pedal-interface
(page 732), side-position-interface (page 738), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.156 VaticanaLigature

A grob to display a melisma (ligature) as used in Gregorian chant. See also KievanLigature
(page 568), MensuralLigature (page 585), and LigatureBracket (page 573).

VaticanaLigature objects are created by: Vaticana_ligature_engraver (page 456).

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): font-interface (page 698),
grob-interface (page 704), spanner-interface (page 746), and vaticana-ligature-
interface (page 762).

This object is of class Spanner (characterized by spanner-interface (page 746)).
3.1.157 **VerticalAlignment**

A top-level auxiliary grob to stack groups (staves, lyrics lines, etc.). See also StaffGrouper (page 630), and VerticalAxisGroup (page 668).

VerticalAlignment objects are created by: Vertical_align_engraver (page 456).

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 #<procedure ly:axis-group-interface::height (_)> #<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): align-interface (page 676), axis-group-interface (page 678), grob-interface (page 704), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.158 **VerticalAxisGroup**

An auxiliary grob to group everything contained in a context like Staff (page 287), Lyrics (page 199), Dynamics (page 126), etc. See also StaffGrouper (page 630), and VerticalAlignment (page 668).

VerticalAxisGroup objects are created by: Axis_group_engraver (page 403).

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 (alist, with symbols as keys):

'((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.

show-vertical-skylines (boolean):

grob::show-skylines-if-debug-skylines-set

If true, print this grob’s vertical skylines. This is meant for debugging purposes.

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 (alist, with symbols as keys):

#<unpure-pure-container #<procedure ly:axis-group-interface::calc-staff-staff-spacing (_)> #<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).
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 #<procedure ly:hara-kiri-group-spanner::y-
   extent (_)> #<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): axis-group-interface (page 678),
   grob-interface (page 704), hara-kiri-group-spanner-interface (page 708),
   outside-staff-axis-group-interface (page 729), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.159 VoiceFollower
A line to indicate staff changes of a voice.

VoiceFollower objects are created by: Note_head_line_engraver (page 438).

Standard settings:
   after-line-breaking (boolean):
      ly:spanner::kill-zero-spanned-time
      Dummy property, used to trigger callback for after-line-breaking.

bound-details (alist, with symbols as keys):
   '((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 (alist, with symbols as keys):
   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 (alist, with symbols as keys):
   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.

This object supports the following interface(s): grob-interface (page 704),
line-interface (page 716), line-spanner-interface (page 717), and spanner-interface
(page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.160 VoltaBracket

A volta bracket. See also VoltaBracketSpanner (page 672).

VoltaBracket objects are created by: Volta_engraver (page 456).

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-features (list):
      '("cv47" "ss01")
      Opentype features.

   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.

\texttt{stencil (stencil):}
\begin{verbatim}
ly:volta-bracket-interface::print
\end{verbatim}
The symbol to print.

\texttt{thickness (number):}
\begin{verbatim}
1.6
\end{verbatim}
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 \texttt{Staff.StaffSymbol.thickness}).

\texttt{vertical-skylines (pair of skylines):}
\begin{verbatim}
#<unpure-pure-container #<procedure ly:grob::vertical-skylines-from-
stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-
extents (_ _ _)> >
\end{verbatim}
Two skylines, one above and one below this grob.

\texttt{word-space (dimension, in staff space):}
\begin{verbatim}
0.6
\end{verbatim}
Space to insert between words in texts.

\texttt{Y-extent (pair of numbers):}
\begin{verbatim}
#<unpure-pure-container #<procedure ly:grob::stencil-height (_)> >
#<procedure volta-bracket-interface::pure-height (grob start end)> >
\end{verbatim}
Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

This object supports the following interface(s): \texttt{font-interface (page 698)}, \texttt{grob-interface (page 704)}, \texttt{horizontal-bracket-interface (page 709)}, \texttt{line-interface (page 716)}, \texttt{side-position-interface (page 738)}, \texttt{spanner-interface (page 746)}, \texttt{text-interface (page 755)}, \texttt{volta-bracket-interface (page 763)}, and \texttt{volta-interface (page 763)}.

This object is of class Spanner (characterized by \texttt{spanner-interface (page 746)}).

\subsection{3.1.161 VoltaBracketSpanner}
An auxiliary grob providing a baseline to align consecutive VoltaBracket (page 671), grobs vertically.

\texttt{VoltaBracketSpanner} objects are created by: \texttt{Volta_engraver (page 456)}.

Standard settings:

\texttt{after-line-breaking (boolean):}
\begin{verbatim}
ly:side-position-interface::move-to-extremal-staff
\end{verbatim}
Dummy property, used to trigger callback for \texttt{after-line-breaking}.

\texttt{axes (list):}
\begin{verbatim}
'(1)
\end{verbatim}
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.

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 #<procedure ly:grob::vertical-skylines-from-
element-stencils (_)> #<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 #<procedure ly:axis-group-interface::height
(_ _)> #<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 #<procedure ly:side-position-interface::y-
aligned-side (_ #:optional _)> #<procedure ly:side-position-
interface::pure-y-aligned-side (_ _ #:optional _)> >
The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): axis-group-interface (page 678),
grob-interface (page 704), outside-staff-interface (page 730), side-position-
interface (page 738), spanner-interface (page 746), and volta-interface (page 763).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.1.162 VowelTransition
A vowel transition in lyrics. See also LyricHyphen (page 575).

VowelTransition objects are created by: Hyphen_ engraver (page 427).
Chapter 3: Backend

Standard settings:

after-line-breaking (boolean):
   ly:spanner::kill-zero-spanned-time
   Dummy property, used to trigger callback for after-line-breaking.

arrow-length (number):
   0.5
   Arrow length.

arrow-width (number):
   0.5
   Arrow width.

bound-details (alist, with symbols as keys):
   '((left (padding . 0.14) (attach-dir . 1))
    (right-broken (padding . 0))
    (left-broken (padding . 0))
    (right (padding . 0.14)
      (attach-dir . -1)
      (arrow . #t)))
   An alist of properties for determining attachments of spanners to edges.

left-bound-info (alist, with symbols as keys):
   ly:horizontal-line-spanner::calc-left-bound-info
   An alist of properties for determining attachments of spanners to edges.

minimum-length (dimension, in staff space):
   1.0
   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.

right-bound-info (alist, with symbols as keys):
   ly:horizontal-line-spanner::calc-right-bound-info
   An alist of properties for determining attachments of spanners to edges.

springs-and-rods (boolean):
   ly:vowel-transition::set-spacing-rods
   Dummy variable for triggering spacing routines.

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 #<procedure ly:grob::vertical-skylines-from-
   stencil (_)> #<procedure ly:grob::pure-simple-vertical-skylines-from-
   extents (_ _ _)> >
   Two skylines, one above and one below this grob.
Y-offset (number):
0.5

The vertical amount that this object is moved relative to its Y-parent.

This object supports the following interface(s): grob-interface (page 704), horizontal-line-spanner-interface (page 710), line-interface (page 716), line-spanner-interface (page 717), lyric-interface (page 720), and spanner-interface (page 746).

This object is of class Spanner (characterized by spanner-interface (page 746)).

3.2 Graphical Object Interfaces

3.2.1 accidental-interface

A single accidental.

User settable properties:

alteration (number)
Alteration numbers for accidental.

alteration-glyph-name-alist (association list (list of pairs))
An alist of key-string pairs.

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.

hide-tied-accidental-after-break (boolean)
If set, an accidental that appears on a tied note after a line break will not be displayed.

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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalSuggestion (page 478), AmbitusAccidental (page 581), and TrillPitchAccidental (page 657).

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 (association list (list of pairs))
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): AccidentalPlacement (page 477).

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): AccidentalSuggestion (page 478).

3.2.4 accidental-switch-interface
Any object that prints one or several accidentals based on alterations.

User settable properties:

alteration-glyph-name-alist (association list (list of pairs))
An alist of key-string pairs.

This grob interface is used in the following graphical object(s): Accidental (page 475), AccidentalCautionary (page 476), AccidentalSuggestion (page 478), AmbitusAccidental (page 481), BalloonText (page 484), BassFigure (page 491), ChordName (page 508), CombineTextScript (page 517), GridChordName (page 553), HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560), KeyCancellation (page 563), KeySignature (page 565), MeasureSpanner (page 583), NoteName (page 597), RehearsalMark (page 607), TEXTScript (page 650), and TrillPitchAccidental (page 657).

3.2.5 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 “NonMusicalPaper-Column” 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 (association list (list of pairs))**
  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): BassFigureAlignment (page 492), and VerticalAlignment (page 668).

### 3.2.6 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): Ambitus (page 479), AmbitusLine (page 481), and AmbitusNoteHead (page 482).

### 3.2.7 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 *(start . end)*, where start and end are vertical positions 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): Arpeggio (page 483).

### 3.2.8 **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)*.

- **nonstaff-nonstaff-spacing** *(alist, with symbols as keys)*
  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 (alist, with symbols as keys)**

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 (alist, with symbols as keys)**

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 (alist, with symbols as keys)**

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-extents 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): Ambitus (page 479),
BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492),
BassFigureLine (page 494), BreakAlignGroup (page 500), BreakAlignment (page 501),
CenteredBarNumberLineSpanner (page 507), DotColumn (page 531), DynamicLineSpanner
(page 538), NonMusicalPaperColumn (page 593), NoteCollision (page 595), NoteColumn
(page 595), PaperColumn (page 600), SostenutoPedalLineSpanner (page 624),
SustainPedalLineSpanner (page 642), System (page 643), TrillPitchGroup (page 658),
UnaCordaPedalLineSpanner (page 666), VerticalAlignment (page 668), VerticalAxisGroup
(page 668), and VoltaBracketSpanner (page 672).

3.2.9 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.
  text-alignment-X (number)
      How to align an annotation horizontally.
  text-alignment-Y (number)
      How to align an annotation vertically.
  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-attachment** (number)
Horizontal attachment of a line on a frame, typically between -1 (left) and 1 (right).

**Y-attachment** (number)
Vertical attachment of a line on a frame, typically between -1 (down) and 1 (up).

**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): BalloonText (page 484),
and Footnote (page 548).

### 3.2.10 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-left (string)
The glyph value to use at the end of the line when the line is broken. #f indicates
that no glyph should be visible; otherwise the value must be a string.

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-right (string)
The glyph value to use at the beginning of the line when the line is broken. #f
indicates that no glyph should be visible; otherwise the value must be a string.
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): BarLine (page 486), and SpanBar (page 626).

3.2.11 bar-number-interface
A bar number or bar number vertical support object.

This grob interface is used in the following graphical object(s): BarNumber (page 489), CenteredBarNumber (page 506), and CenteredBarNumberLineSpanner (page 507).

3.2.12 bass-figure-alignment-interface
Align a bass figure.

This grob interface is used in the following graphical object(s): BassFigureAlignment (page 492).

3.2.13 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): BassFigure (page 491).

3.2.14 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 $\leq$ `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:**

- **auto-knee-gap** (dimension, in staff space)
  If a gap is found between note heads where a horizontal beam fits and it 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 collision 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 (alist, with symbols as keys)
  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 position to a (quantized) position that is as close as possible to this value, and print the demerits for the inspected position in the output.

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 (start . end), where start and end are vertical positions 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:

- **annotation** (string)
  Annotate a grob for debug purposes.

- **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): Beam (page 495).

### 3.2.15 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): BendAfter (page 497).

### 3.2.16 bend-interface

The (curved) line representing a bent string. Available for the 'style' property are 'hold', 'pre-bend' and 'pre-bend-hold. The following properties may be set in the details list.

- **arrow-stencil**
  The stencil procedure for the BendSpanner arrow head.

- **curvature-factor**
  Determines the horizontal part of a bend arrow as percentage of the total horizontal extent, usually between 0 and 1.

- **bend-arrowhead-height**
  The height of the arrow head.
bend-arrowhead-width
The width of the arrow head.

bend-amount-strings
An alist with entries for 'quarter, 'half, 'three-quarter and 'full, which are used to print how much a string is bent.

curve-x-padding-line-end
For a broken BendSpanner, set the padding at the line end to subsequent objects like changed Clef, etc.

curve-y-padding-line-end
For a broken BendSpanner started from a chord the curves don’t match; there is a certain vertical gap specified by this value.

dashed-line-settings
List of three numeric values representing on, off and phase of a dashed line.

head-text-break-visibility
A vector of three booleans to set visibility of the arrow head and the text at a line break.
This is important for 'style set to 'hold, 'pre-bend or 'pre-bend-hold.

horizontal-left-padding
The amount of horizontal free space between a TabNoteHead and the starting BendSpanner.

successive-level
An integer used as a factor determining the vertical coordinate of the starting BendSpanner.
If successive-level is 1, the BendSpanner starts at the TabNoteHead. If consecutive BendSpanners are set this value should be set to an appropriate value for the first one; later on, this value is maintained by the engraver.

target-visibility
A boolean to decide whether the target TabNoteHead should be visible. For up-pointing bends this is usually true.

y-distance-from-tabstaff-to-arrow-tip
This numeric value determines the distance between the TabStaff and the arrow head of the BendSpanner.

User settable properties:

bend-me (boolean)
Decide whether this grob is bent.

details (alist, with symbols as keys)
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.

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): BendSpanner (page 498), NoteColumn (page 595), NoteHead (page 596), and TabNoteHead (page 648).
3.2.17 bezier-curve-interface

A Bézier curve (tie, slur, etc.).

**User settable properties:**

- `show-control-points` (boolean)
  For grobs printing Bézier curves, setting this property to true causes the control points and control polygon to be drawn on the page for ease of tweaking.

This grob interface is used in the following graphical object(s): LaissezVibrerTie (page 569), PhrasingSlur (page 604), RepeatTie (page 610), Slur (page 621), and Tie (page 653).

3.2.18 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): BarNumber (page 489), CodaMark (page 515), JumpScript (page 561), LyricRepeatCount (page 576), MetronomeMark (page 585), RehearsalMark (page 607), SectionLabel (page 615), and SegnoMark (page 616).

3.2.19 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` (alist, with symbols as keys)
  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 [Wanské] page 126–134, [Ross] page 143–147.

This grob interface is used in the following graphical object(s): Ambitus (page 479), AmbitusAccidental (page 481), BarLine (page 486), BreakAlignGroup (page 500), BreathingSign (page 502), Clef (page 510), CueClef (page 521), CueEndClef (page 524), Custos (page 527), Divisio (page 528), DoublePercentRepeat (page 532), KeyCancellation (page 563), KeySignature (page 565), LeftEdge (page 571), SignumRepetitionis (page 618), SpanBar (page 620), StaffEllipsis (page 628), and TimeSignature (page 655).

### 3.2.20 break-alignment-interface

The object that performs break alignment.

Three interfaces deal specifically with break alignment:

1. break-alignment-interface (this one),
2. Section 3.2.18 [break-alignable-interface], page 687, and
3. Section 3.2.19 [break-aligned-interface], page 687.
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
- signum-repetitionis
- staff-bar
- staff-ellipsis
- 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:

  ```latex
  \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): BreakAlignment (page 501).

**3.2.21 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.

**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): BreathingSign (page 502), and Divisio (page 528).

3.2.22 **caesura-script-interface**
A script for \caesura, e.g., an outside-staff comma or a fermata over a bar line.

This grob interface is used in the following graphical object(s): CaesuraScript (page 504).

3.2.23 **centered-bar-number-interface**
A measure-centered bar number.

This grob interface is used in the following graphical object(s): CenteredBarNumber (page 506).

3.2.24 **centered-bar-number-line-spanner-interface**
An abstract object used to align centered bar numbers on the same vertical position.

This grob interface is used in the following graphical object(s): CenteredBarNumberLineSpanner (page 507).

3.2.25 **centered-spanner-interface**
A spanner that prints a symbol centered between two columns.

User settable properties:

**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.

**spacing-pair** (pair)
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)

This grob interface is used in the following graphical object(s): CenteredBarNumber (page 506), MeasureCounter (page 580), and PercentRepeat (page 602).
3.2.26 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): ChordName (page 508), and FretBoard (page 550).

3.2.27 chord-square-interface
A chord square in a chord grid.

**User settable properties:**

- `measure-division` (number list)
  - A list representing what fraction of the measure length each chord name takes in a chord square. The list is made of exact numbers between 0 and 1, which should add up to 1. Example: a measure c2 g4 g4 results in '/(1/2 1/4 1/4).

- `measure-division-chord-placement-alist` (association list (list of pairs))
  - An alist mapping measure divisions (see the measure-division property) to lists of coordinates (number pairs) applied to the chord names of a chord square. Coordinates are normalized between -1 and 1 within the square.

- `measure-division-lines-alist` (association list (list of pairs))
  - An alist mapping measure divisions (see the measure-division property) to lists of lines to draw in the square, given as 4-element lists: `(x-start y-start x-end y-end)`.

**Internal properties:**

- `chord-names` (array of grobs)
  - Array of chord names.

This grob interface is used in the following graphical object(s): ChordSquare (page 509).

3.2.28 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): Clef (page 510), CueClef (page 521), and CueEndClef (page 524).

3.2.29 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 (alist, with symbols as keys)
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): ClefModifier (page 513).

3.2.30 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 (start, end), where start and end are vertical positions 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): ClusterSpannerBeacon (page 515).

3.2.31 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): ClusterSpanner (page 514).

3.2.32 coda-mark-interface
A coda sign.

This grob interface is used in the following graphical object(s): CodaMark (page 515).
3.2.33 control-point-interface
A grob used to visualize one control point of a Bézier curve (such as a tie or a slur), for ease of tweaking.

Internal properties:

bezier (graphical (layout) object)
A pointer to a Bézier curve, for use by control points and polygons.

index (non-negative, exact integer)
For some grobs in a group, this is a number associated with the grob.

This grob interface is used in the following graphical object(s): ControlPoint (page 519).

3.2.34 control-polygon-interface
A grob used to visualize the control polygon of a Bézier curve (such as a tie or a slur), for ease of tweaking.

User settable properties:

extroversion (number)
For polygons, how the thickness of the line is spread on each side of the exact polygon with ideal zero thickness. If this is 0, the middle of line is on the polygon. If 1, the line sticks out of the polygon. If -1, the outer side of the line is exactly on the polygon. Other numeric values are interpolated.

filled (boolean)
Whether an object is filled with ink.

Internal properties:

bezier (graphical (layout) object)
A pointer to a Bézier curve, for use by control points and polygons.

This grob interface is used in the following graphical object(s): ControlPolygon (page 520).

3.2.35 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): Custos (page 527).

3.2.36 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): DotColumn (page 531).

3.2.37 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): Dots (page 531).

3.2.38 duration-line-interface
A line lasting for the duration of a rhythmic event.

User settable properties:

details (alist, with symbols as keys)
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.

This grob interface is used in the following graphical object(s): DurationLine (page 536).
3.2.39 **dynamic-interface**

Any kind of loudness sign.

This grob interface is used in the following graphical object(s): DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541), and Hairpin (page 555).

3.2.40 **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): DynamicLineSpanner (page 538).

3.2.41 **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): DynamicText (page 539).

3.2.42 **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): DynamicTextSpanner (page 541).

3.2.43 **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.

- **dashed-edge** (boolean)
  If set, the bracket edges are dashed like the rest of the bracket.
edge-height (pair)
   A pair of numbers specifying the heights of the vertical edges: \((left\text{-}height, right\text{-}height)\).

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 \textit{Staff.StaffSymbol.thickness}).

\textbf{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): \textit{BassFigureBracket} (page 493).

3.2.44 \textit{episema-interface}

An episema line.

   This grob interface is used in the following graphical object(s): \textit{Episema} (page 542).

3.2.45 \textit{figured-bass-continuation-interface}

Simple extender line between bounds.

\textbf{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 \textit{Staff.StaffSymbol.thickness}).

\textbf{Internal properties:}

   figures (array of grobs)
      Figured bass objects for continuation line.

   This grob interface is used in the following graphical object(s): \textit{BassFigureContinuation} (page 494).

3.2.46 \textit{finger-glide-interface}

The line between \textit{Fingering} grobs indicating a glide with that finger.
The property style may take the following symbols.

**line**
A simple connecting line.

**dashed-line**
Print a dashed line. Customizable with settings for dash-fraction and dash-period.

**dotted-line**
Print a dotted line.

**stub-right**
The printed line is limited to a certain amount right before its right bound. This amount is configurable by a suitable setting for bound-details.right.right-stub-length.

**stub-left**
The printed line is limited to a certain amount right after its left bound. The amount is configurable by a suitable setting for bound-details.right.left-stub-length.

**stub-both**
The printed line combines the settings of stub-left and stub-right.

**zigzag**
A zigzag line, configurable with suitable settings for zigzag-width and zigzag-length.

**trill**
A trill style line.

**bow**
A bow style line. The orientation of the bow may be tweaked with a suitable setting of details.bow-direction.

**User settable properties:**

- **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.

- **details (alist, with symbols as keys)**
  A list 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.

- **style (symbol)**
  This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

- **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 spanner line can be constructed from a whole number of squiggles.

This grob interface is used in the following graphical object(s): FingerGlideSpanner (page 543).
3.2.47 **finger-interface**

A fingering instruction.

This grob interface is used in the following graphical object(s): Fingering (page 545).

3.2.48 **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): FingeringColumn (page 547).

3.2.49 **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): Flag (page 547).

3.2.50 **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): Accidental (page 475), AccidentalCautionary (page 476), AccidentalSuggestion (page 478), AmbitusAccidental (page 481), AmbitusLine (page 481), AmbitusNoteHead (page 482), Arpeggio (page 483), BalloonText (page 484), BarLine (page 486), BarNumber (page 489), BassFigure (page 491), BendSpanner (page 498), BreathingSign (page 502), CaesuraScript (page 504), CenteredBarNumber (page 506), ChordName (page 508), Clef (page 510), ClefModifier (page 513), CodaMark (page 515), CombineTextScript (page 517), CueClef (page 521), CueEndClef (page 524), Custos (page 527), Divisio (page 528), Dots (page 531), DoublePercentRepeat (page 532), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DurationLine (page 536), DynamicText (page 539), DynamicTextSpanner (page 541), Episema (page 542), Fingering (page 545), Flag (page 547), Footnote (page 548), FretBoard (page 550), GridChordName (page 553), HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560), JumpScript (page 561), KeyCancellation (page 563), KeySignature (page 565), KievanLigature (page 568), LyricHyphen (page 575), LyricRepeatCount (page 576), LyricText (page 579), MeasureCounter (page 580), MeasureSpanner (page 583), MensuralLigature (page 585), MetronomeMark (page 585), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), NonMusicalPaperColumn (page 593), NoteHead (page 596), NoteName (page 597), OttavaBracket (page 599), PaperColumn (page 600), Parentheses (page 601), PercentRepeat (page 602), PercentRepeatCounter (page 603), RehearsalMark (page 607), Rest (page 611), Script (page 613), SectionLabel (page 615), SegnoMark (page 616), SignumRepetitionis (page 618), SostenutoPedal (page 623), SpanBar (page 626), StaffEllipsis (page 628), StanzNumber (page 633), StringNumber (page 638), StrokeFinger (page 640), SustainPedal (page 641), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), TabNoteHead (page 648), TextScript (page 650), TextSpanner (page 652), TimeSignature (page 655), TrillPitchAccidental (page 657), TrillPitchHead (page 659), TrillPitchParentheses (page 661).
(page 660), TrillSpanner (page 661), TupletNumber (page 664), UnaCordaPedal (page 665), VaticanaLigature (page 667), and VoltaBracket (page 671).

### 3.2.51 footnote-interface

Make a footnote.

**User settable properties:**

- automatically-numbered (boolean)
  - If set, footnotes are 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.
- 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): Footnote (page 548).

### 3.2.52 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 (alist, with symbols as keys)
  - 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.

• handedness – Print the fret-diagram left- or right-handed. -1, LEFT for left ; 1, RIGHT for right. Default RIGHT.

• 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 arabic, roman-ij-lower, roman-ij-upper, roman-lower, roman-upper, arabic and custom. In the last 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.

\textbf{size (number)}

The ratio of the size of the object to its default size.
Chapter 3: Backend

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): FretBoard (page 550).

3.2.53 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): Glissando (page 552).

3.2.54 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): GraceSpacing (page 553).

3.2.55 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 diminished?

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): NoteHead (page 596).

3.2.56 grid-chord-name-interface
A chord name in a chord grid.

Internal properties:

index (non-negative, exact integer)
    For some grobs in a group, this is a number associated with the grob.

This grob interface is used in the following graphical object(s): GridChordName (page 553).

3.2.57 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): GridLine (page 554).

3.2.58 grid-point-interface
A spanning point for grid lines.

This grob interface is used in the following graphical object(s): GridPoint (page 555).
3.2.59 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 3.1.93 [NoteCollision], page 595, 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 (association list (list of pairs))
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"))
produces
 <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 has 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.

parenthesis-id (symbol)
When parenthesized grobs created in the same time step have this property, there is
one set of parentheses for each group of grobs having the same value.

parenthesized (boolean)
Parenthesize this grob.

rotation (list)
Number of degrees to rotate this object, and what point to rotate around. For exam-
ple, '(45 0 0) rotates by 45 degrees around the center of this object.

show-horizontal-skylines (boolean)
If true, print this grob’s horizontal skylines. This is meant for debugging purposes.

show-vertical-skylines (boolean)
If true, print this grob’s vertical skylines. This is meant for debugging purposes.

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 (alist, with symbols as keys)
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):
Accidental (page 475), AccidentalCautionary (page 476), AccidentalPlacement (page 477),
AccidentalSuggestion (page 478), Ambitus (page 479), AmbitusAccidental (page 481),
AmbitusLine (page 481), AmbitusNoteHead (page 482), Arpeggio (page 483), BalloonText (page 484),
BarLine (page 486), BarNumber (page 489), BassFigure (page 491),
BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492),
BassFigureBracket (page 493), BassFigureContinuation (page 494), BassFigureLine (page 494),
Beam (page 495), BendAfter (page 497), BendSpanner (page 498),
BreakAlignGroup (page 500), BreakAlignment (page 501), BreathingSign (page 502),
Caesura SCRIPT (page 504), CenteredBarNumber (page 506), CenteredBarNumberLineSpanner (page 507),
ChordName (page 508), ChordSquare (page 509), Clef (page 510), ClefModifier (page 513),
ClusterSpanner (page 514), ClusterSpannerBeacon (page 515), CodaMark (page 515),
CombineTextScript (page 517), ControlPoint (page 519), ControlPolygon (page 520),
CueClef (page 521), CueEndClef (page 524), Custos (page 527), Divisio (page 528),
DotColumn (page 531), Dots (page 531), DoublePercentRepeat (page 532),
DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DurationLine (page 536),
DynamicLineSpanner (page 538), DynamicText (page 539), DynamicTextSpanner (page 541),
Episema (page 542), FingerGlideSpanner (page 543), Fingering (page 545),
FingeringColumn (page 547), Flag (page 547), Footnote (page 548), FretBoard (page 550),
Glissando (page 552), GraceSpacing (page 553), GridChordName (page 554),
GridLine (page 554), GridPoint (page 555), Hairpin (page 555),
HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560),
JumpScript (page 561), KeyCancellation (page 563), KeySignature (page 565),
KiervanLigature (page 568), LaissezVibrerTie (page 569), LaissezVibrerTieColumn (page 570),
LedgerLineSpanner (page 570), LeftEdge (page 571), LigatureBracket (page 573),
LyricExtender (page 574), LyricHyphen (page 575), LyricRepeatCount (page 576),
LyricSpace (page 578), LyricText (page 579), MeasureCounter (page 580),
MeasureGrouping (page 582), MeasureSpanner (page 583), MelodyItem (page 585),
MensuralLigature (page 585), MetronomeMark (page 585), MultiMeasureRest (page 587),
MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590),
MultiMeasureRestText (page 592), NonMusicalPaperColumn (page 593), NoteCollision (page 595),
NoteColumn (page 595), NoteName (page 596), NoteName (page 597),
NoteSpanning (page 598), OttavaBracket (page 599), PaperColumn (page 600), Parentheses (page 601),
PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604),
PianoPedalBracket (page 606), RehearsalMark (page 607), RepeatSlash (page 609),
RepeatTie (page 610), RepeatTieColumn (page 611), Rest (page 611),
RestCollision (page 612), Script (page 613), ScriptColumn (page 614), ScriptRow (page 614),
SectionLabel (page 615), SegnoMark (page 616), SignumRepetitionis (page 618),
Slur (page 621), SostenutoPedal (page 623), SostenutoPedalLineSpanner (page 624),
SpacingSpanner (page 626), SpanBar (page 626), SpanBarStub (page 628),
StaffEllipsis (page 628), StaffGrouper (page 630), StaffHighlight (page 631),
StaffSpacing (page 632), StaffSymbol (page 632), StanzaNumber (page 633), Stem (page 634),
StemStub (page 636), StemTremolo (page 637), StringNumber (page 638),
StrokeFinger (page 640), SustainPedal (page 641), SustainPedalLineSpanner (page 642),
System (page 643), SystemStartBar (page 644), SystemStartBrace (page 645),
SystemStartBracket (page 646), SystemStartSquare (page 647), TabNoteHead (page 648),
TextScript (page 650), TextSpanner (page 652), Tie (page 653), TieColumn (page 655),
3.2.60 **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).

- **endpoint-alignments (pair of numbers)**
  A pair of numbers representing the alignments of an object’s endpoints. E.g., the ends of a hairpin relative to NoteColumn grobs.

- **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): Hairpin (page 555).

3.2.61 **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 VerticalAxisGroups. If any of them are alive, then we will stay alive.
- make-dead-when (array of grobs)
  An array of other VerticalAxisGroups. If any of them are alive, then we will turn dead.

This grob interface is used in the following graphical object(s): VerticalAxisGroup (page 668).

3.2.62 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.
- dashed-edge (boolean)
  If set, the bracket edges are dashed like the rest of the bracket.
- 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): HorizontalBracket (page 557), OttavaBracket (page 599), and VoltaBracket (page 671).
3.2.63 **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): HorizontalBracketText (page 558).

3.2.64 **horizontal-line-spanner-interface**

This interface is a subset of the Section 3.2.78 [line-spanner-interface], page 717, for use with line spanners that are always horizontal (such as crescendo spanners). The details.Y subproperty is irrelevant. Grobs having this interface can be side-positioned vertically.

This grob interface is used in the following graphical object(s): DurationLine (page 536), DynamicTextSpanner (page 541), Episema (page 542), TextSpanner (page 652), TrillSpanner (page 661), and VowelTransition (page 673).

3.2.65 **inline-accidental-interface**

An inlined accidental (i.e., normal accidentals, cautionary accidentals).

This grob interface is used in the following graphical object(s): Accidental (page 475), AccidentalCautionary (page 476), and TrillPitchAccidental (page 657).

3.2.66 **instrument-specific-markup-interface**

Instrument-specific markup (like fret boards or harp pedal diagrams).

**User settable properties:**

- **fret-diagram-details** (alist, with symbols as keys)
  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 to 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.
• **handedness** – Print the fret-diagram left- or right-handed. −1, LEFT for left; 1, RIGHT for right. Default RIGHT.
• **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 arabic, roman-ij-lower, roman-ij-upper, roman-lower, roman-upper, arabic and custom. In the last 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 \( \text{thickness} \times (1 + \text{string-thickness-factor}) \times (k-1) \). 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 (alist, with symbols as keys)
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).

woodwind-diagram-details (alist, with symbols as keys)
An alist of detailed grob properties for woodwind diagrams. Each alist entry consists of a (property . value) pair. The properties which can be included in woodwind-diagram-details include the following:
• fill-angle – Rotation angle of a partially filled key from horizontal. Default value 0.
• text-trill-circled – In non-graphical mode, for keys shown as text, indicate a trill by circling the text if true, or by shading the text if false. Default value #t.

This grob interface is used in the following graphical object(s): TextScript (page 650).

3.2.67 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:

<table>
<thead>
<tr>
<th>grob will show:</th>
<th>before</th>
<th>no</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-invisible</td>
<td>break</td>
<td>break</td>
<td>break</td>
</tr>
<tr>
<td>begin-of-line-visible</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
User settable properties:

break-visibility (vector)
A vector of 3 booleans, #$\langle\text{end-of-line unbroken begin-of-line}\rangle$. #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.
TrillPitchAccidental (page 657), TrillPitchGroup (page 658), TrillPitchHead (page 659), TrillPitchParentheses (page 660), and UnaCordaPedal (page 665).

In addition, this interface is supported conditionally by the following objects depending on their class: BalloonText (page 484), ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), and Parentheses (page 601).

3.2.68 jump-script-interface
A jump instruction, e.g., D.S.
This grob interface is used in the following graphical object(s): JumpScript (page 561).

3.2.69 key-cancellation-interface
A key cancellation.
This grob interface is used in the following graphical object(s): KeyCancellation (page 563).

3.2.70 key-signature-interface
A group of accidentals, to be printed as signature sign.

User settable properties:

- alteration-alist (association list (list of pairs))
  List of (pitch . accidental) pairs for key signature.
- alteration-glyph-name-alist (association list (list of pairs))
  An alist of key-string pairs.
- 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.
- 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 (association list (list of pairs))
  An alist of padding pairs for key signatures (and key cancellations). Each alist entry has the form
  
  ((left-glyph-name . right-glyph-name) . dist)
  specifying the padding dist between two adjacent key signature elements. If there is no entry in the alist for a given pair, the padding value given by the padding property of the KeySignature (or KeyCancellation) grob is used instead.
  A special feature is the handling of adjacent naturals (to be more precise, the handling of glyph accidentals.natural): If there is no ‘natural-natural’ entry in padding-pairs explicitly overriding it, LilyPond adds some extra padding (in addition to the grob’s padding value) to avoid collisions.
- 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): KeyCancellation (page 563), and KeySignature (page 565).

### 3.2.71 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): KievanLigature (page 568).

### 3.2.72 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 3.1.127 [StaffSymbol], page 632, 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): LedgerLineSpanner (page 570).

### 3.2.73 ledgered-interface

Objects that need ledger lines, typically note heads. See also Section 3.2.72 [ledger-line-spanner-interface], page 715.
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): AmbitusNoteHead (page 482), NoteHead (page 596), and TrillPitchHead (page 659).

3.2.74 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.

3.2.75 ligature-head-interface
A note head that can become part of a ligature.

This grob interface is used in the following graphical object(s): NoteHead (page 596).

3.2.76 ligature-interface
A ligature.

This grob interface is not used in any graphical object.

3.2.77 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.
Chapter 3: Backend

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 spanner
line can be constructed from a whole number of squiggles.

This grob interface is used in the following graphical object(s): ChordSquare (page 509),
DurationLine (page 536), DynamicTextSpanner (page 541), Episema (page 542), Glissando
(page 552), Hairpin (page 555), HorizontalBracket (page 557), LigatureBracket
(page 573), MeasureSpanner (page 583), OttavaBracket (page 599), PianoPedalBracket
(page 606), TextSpanner (page 652), TrillSpanner (page 661), TupletBracket (page 662),
VoiceFollower (page 670), VoltaBracket (page 671), and VowelTransition (page 673).

3.2.78 line-spanner-interface
Generic line drawn between two objects, e.g., for use with glissandi.

bound-details is a nested alist. It’s possible to specify settings for the sub-properties: left,
left-broken, right and right-broken.

Values for the following keys may be set:

Y  Sets the Y coordinate of the end point, in staff-spaces offset from the staff center line. By
default, it is the center of the bound object, so a glissando points to the vertical center of the
note head. Not relevant for grobs having the Section 3.2.64 [horizontal-line-spanner-
interface], page 710.

attach-dir
Determines where the line starts and ends in the X direction, relative to the bound object.
So, a value of -1 (or LEFT) makes the line start/end at the left side of the note head it is
attached to.

X  This is the absolute X coordinate of the end point. Usually computed on the fly.

end-on-note
If set to true, when the line spanner is broken, each broken piece only extends to the furthest
note, not to the end of the staff, on sides where it is broken.

end-on-accidental
Only meaningful in bound-details.right. If set to true, the line spanner stops before the
accidentals if its right bound is a note column or a grob contained in a note column, and
this note column has accidentals.

start-at-dot
Only meaningful in bound-details.left. If true, the line spanner starts after dots, in a
fashion similar to end-on-accidental.
adjust-on-neighbor
If true, the left-neighbor or right-neighbor object is read, and if it exists, the line spanner starts after it or stops before it.

stencil
Line spanners may have symbols at the beginning or end, which is contained in this sub-property. For internal use.

text
This is a markup that is evaluated to yield the stencil.

stencil-align-dir-y
stencil-offset
Without setting one of these, the stencil is simply put at the end-point, centered on the line, as defined by the X and Y sub-properties. Setting stencil-align-dir-y moves the symbol at the edge vertically relative to the end point of the line. With stencil-offset, expecting a number pair, the stencil is moved along the X axis according to the first value, the second value moves the stencil along the Y axis.

arrow
Produces an arrowhead at the end-points of the line.

padding
Controls the space between the specified end point of the line and the actual end. Without padding, a glissando would start and end in the center of each note head.

User settable properties:

bound-details (alist, with symbols as keys)
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 (alist, with symbols as keys)
An alist of properties for determining attachments of spanners to edges.

right-bound-info (alist, with symbols as keys)
An alist of properties for determining attachments of spanners to edges.

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:

left-neighbor (graphical (layout) object)
A grob similar to this one, on its left. For columns, the right-most column that has a spacing wish for this column.

note-columns (array of grobs)
An array of NoteColumn grobs.
right-neighbor (graphical (layout) object)
  See left-neighbor.

This grob interface is used in the following graphical object(s): BendSpanner (page 498),
DurationLine (page 536), DynamicTextSpanner (page 541), Episema (page 542),
FingerGlideSpanner (page 543), Glissando (page 552), TextSpanner (page 652),
TrillSpanner (page 661), VoiceFollower (page 670), and VowelTransition (page 673).

3.2.79 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): LyricExtender (page 574).

3.2.80 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 (each unit represents half a
  staff-space).

  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 \texttt{Staff.StaffSymbol.thickness}).

This grob interface is used in the following graphical object(s): LyricHyphen (page 575),
and LyricSpace (page 578).

3.2.81 lyric-interface
Any object that is related to lyrics.

This grob interface is used in the following graphical object(s): LyricExtender (page 574),
LyricHyphen (page 575), LyricRepeatCount (page 576), and VowelTransition (page 673).

3.2.82 lyric-repeat-count-interface
A repeat count intended to appear in a line of lyrics.

This grob interface is used in the following graphical object(s): LyricRepeatCount
(page 576).

3.2.83 lyric-space-interface
An invisible object that prevents lyric words from being spaced too closely.

This grob interface is used in the following graphical object(s): LyricSpace (page 578).

3.2.84 lyric-syllable-interface
A single piece of lyrics.

This grob interface is used in the following graphical object(s): LyricText (page 579).

3.2.85 mark-interface
A rehearsal mark, segno, or coda sign.

This grob interface is used in the following graphical object(s): CodaMark (page 515),
RehearsalMark (page 607), and SegnoMark (page 616).

3.2.86 measure-counter-interface
A counter for numbering measures.

User settable properties:

\texttt{count-from} (integer)
The first measure in a measure count receives this number. The following measures
are numbered in increments from this initial value.

\texttt{left-number-text} (markup)
For a measure counter, this is the formatted measure count. When the measure
counter extends over several measures (like with compressed multi-measure rests), it
is the text on the left side of the dash.

\texttt{number-range-separator} (markup)
For a measure counter extending over several measures (like with compressed multi-
measure rests), this is the separator between the two printed numbers.
right-number-text (markup)
When the measure counter extends over several measures (like with compressed multi-
measure rests), this is the text on the right side of the dash. Usually unset.

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): MeasureCounter
(page 580).

3.2.87 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): MeasureGrouping
  (page 582).

3.2.88 measure-spanner-interface
A bracket aligned to a measure or measures.

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.

  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.

  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).

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.

spacing-pair (pair)
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)
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).

This grob interface is used in the following graphical object(s): MeasureSpanner (page 583).

3.2.89 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): MelodyItem (page 585).

3.2.90 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 \texttt{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): \texttt{MensuralLigature} (page 585), and \texttt{NoteHead} (page 596).

### 3.2.91 metronome-mark-interface

A metronome mark.

This grob interface is used in the following graphical object(s): \texttt{MetronomeMark} (page 585).

### 3.2.92 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): \texttt{MultiMeasureRest} (page 587), \texttt{MultiMeasureRestNumber} (page 589), \texttt{MultiMeasureRestScript} (page 590), and \texttt{MultiMeasureRestText} (page 592).

### 3.2.93 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 \textit{not} influenced by changes to \texttt{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
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)

thick-thickness (number)
Thickness of the thick line in a bar line, expressed as a multiple of the de-
fault staff-line thickness (i.e., the visual output is not influenced by changes to
Staff.StaffSymbol.thickness).

usable-duration-logs (list)
List of duration-logs 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): MultiMeasureRest
(page 587), and PercentRepeat (page 602).

3.2.94 multi-measure-rest-number-interface
Multi measure rest number that is printed over a rest.

This grob interface is used in the following graphical object(s): MultiMeasureRestNumber
(page 589).

3.2.95 musical-paper-column-interface
A paper column that is musical. Paper columns of this variety group musical items, such as
note heads, stems, dots, accidentals, ...
User settable properties:

*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.

Internal properties:

*grace-spacing* (graphical (layout) object)
A run of grace notes.

This grob interface is used in the following graphical object(s): PaperColumn (page 600).

### 3.2.96 non-musical-paper-column-interface

A paper column that is non-musical. Paper columns of this variety group breakable items such as clefs, bar lines, time signatures, key signatures, breathing signs, . . .

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.

*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* (alist, with symbols as keys)
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.
Internal properties:

- break-alignment (graphical (layout) object)
  The BreakAlignment (page 501), in a NonMusicalPaperColumn (page 593).

This grob interface is used in the following graphical object(s): NonMusicalPaperColumn (page 593).

3.2.97 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 3.2.98 [note-column-interface], page 726: these are force-hshift and horizontal-shift.

User settable properties:

- fa-merge-direction (direction)
  If two ‘fa’ shape note heads get merged that are both listed in the fa-styles property but have different stem directions, enforce this note head direction for display.

- 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:

- fa-styles (symbol list)
  List of note head styles that identify ‘fa’ shape note heads.

- 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): NoteCollision (page 595).

3.2.98 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 NoteColumns 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.

main-extent (pair of numbers)
The horizontal extent of a NoteColumn grob without taking suspended NoteHead grobs into account (i.e., NoteHeads forced into the unnatural direction of the Stem because of a chromatic clash).

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): NoteColumn (page 595).

3.2.99 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): AmbitusNoteHead
      (page 482), NoteHead (page 596), and TabNoteHead (page 648).

3.2.100 note-name-interface

Note names.

   This grob interface is used in the following graphical object(s): NoteName (page 597).

3.2.101 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): NoteSpacing (page 598).

3.2.102 number-interface

Numbers.
User settable properties:

number-type (symbol)
Numbering style. Choices include arabic, roman-ij-lower, roman-ij-upper, roman-lower, and roman-upper.

This grob interface is used in the following graphical object(s): StringNumber (page 638).

3.2.103 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.

dashed-edge (boolean)
If set, the bracket edges are dashed like the rest of the bracket.

dashed-edge (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): OttavaBracket (page 599).

3.2.104 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): BassFigureLine (page 494), System (page 643), and VerticalAxisGroup (page 668).
3.2.105 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 it is 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): AccidentalSuggestion (page 478), BarNumber (page 489), BassFigureAlignmentPositioning (page 492), BendSpanner (page 498), BreathingSign (page 502), CaesuraScript (page 504), CenteredBarNumberLineSpanner (page 507), ChordName (page 508), ClefModifier (page 513), CodaMark (page 515), CombineTextScript (page 517), Divisio (page 528), DoublePercentRepeatCounter (page 533), DoubleRepeatSlash (page 535), DynamicLineSpanner (page 538), DynamicText (page 539), Fingering (page 545), FretBoard (page 550), Hairpin (page 555), HorizontalBracket (page 557), HorizontalBracketText (page 558), InstrumentSwitch (page 560), JumpScript (page 561), MeasureCounter (page 580), MeasureGrouping (page 582), MeasureSpanner (page 583), MetronomeMark (page 585), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), OttavaBracket (page 599), PercentRepeatCounter (page 603), PhrasingSlur (page 604), RehearsalMark (page 607), Script (page 613), SectionLabel (page 615), SegnoMark (page 616), Slur (page 621), SostenutoPedalLineSpanner (page 624), StringNumber (page 638), StrokeFinger (page 640), SustainPedalLineSpanner (page 642), TextScript (page 650), TextSpanner (page 652), TritiSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), UnaCordaPedalLineSpanner (page 666), and VoltaBracketSpanner (page 672).

3.2.106 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.

The paper-column-interface implies the item-interface (page 712).

User settable properties:

labels (list)
List of labels (symbols) placed on a column.
rhythmic-location (rhythmic location)
Where (bar number, measure position) in the score.

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.

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): NonMusicalPaperColumn
(page 593), and PaperColumn (page 600).

3.2.107 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): Parentheses (page 601),
and TrillPitchParentheses (page 660).

3.2.108 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): DoublePercentRepeat (page 532), DoubleRepeatSlash (page 535), PercentRepeat (page 602), and RepeatSlash (page 609).

3.2.109 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.

dashed-edge (boolean)
If set, the bracket edges are dashed like the rest of the bracket.

dashed-edge (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): PianoPedalBracket (page 606).

3.2.110 piano-pedal-interface
A piano pedal sign.

This grob interface is used in the following graphical object(s): PianoPedalBracket (page 606), SostenutoPedalLineSpanner (page 624), SustainPedal (page 641), SustainPedalLineSpanner (page 642), and UnaCordaPedalLineSpanner (page 666).

3.2.111 piano-pedal-script-interface
A piano pedal sign, fixed size.

This grob interface is used in the following graphical object(s): SostenutoPedal (page 623), SustainPedal (page 641), and UnaCordaPedal (page 665).

3.2.112 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): TrillPitchHead (page 659), and TrillPitchParentheses (page 660).
3.2.113 **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): BarLine (page 486), Clef (page 510), CueClef (page 521), CueEndClef (page 524), KeyCancellation (page 563), KeySignature (page 565), SignumRepetitionis (page 618), SpanBarStub (page 628), and TimeSignature (page 655).

3.2.114 **rehearsal-mark-interface**

A rehearsal mark.

This grob interface is used in the following graphical object(s): RehearsalMark (page 607).

3.2.115 **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): RestCollision (page 612).

3.2.116 **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): MultiMeasureRest
(page 587), and Rest (page 611).

3.2.117 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): BassFigure (page 491),
ChordName (page 508), ClusterSpannerBeacon (page 515), DoubleRepeatSlash (page 535),
FretBoard (page 550), LyricText (page 579), NoteHead (page 596), RepeatSlash (page 609),
Rest (page 611), and TabNoteHead (page 648).

3.2.118 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): AmbitusNoteHead
(page 482), NoteHead (page 596), Rest (page 611), TabNoteHead (page 648), and
TrillPitchHead (page 659).

3.2.119 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): ScriptColumn (page 614),
and ScriptRow (page 614).

3.2.120 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): AccidentalSuggestion (page 478), CaesuraScript (page 504), DynamicText (page 539), MultiMeasureRestScript (page 590), and Script (page 613).

3.2.121 section-label-interface
A section label, e.g., “Coda”.

This grob interface is used in the following graphical object(s): SectionLabel (page 615).
3.2.122 segno-mark-interface
A segno.

This grob interface is used in the following graphical object(s): SegnoMark (page 616).

3.2.123 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): AccidentalSuggestion (page 478), BarNumber (page 489), CaesuraScript (page 504), ClefModifier (page 513), CodaMark (page 515), CombineTextScript (page 517), DoublePercentRepeatCounter (page 533), DynamicText (page 539), Fingering (page 545), GridLine (page 554), Hairpin (page 555), HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560), JumpScript (page 561), LyricRepeatCount (page 576), LyricText (page 579), MeasureCounter (page 580), MeasureSpanner (page 583), MetronomeMark (page 585), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), PercentRepeatCounter (page 603), RehearsalMark (page 607), Script (page 613), SectionLabel (page 615), SegnoMark (page 616), SostenutoPedal (page 623), StemTremolo (page 637), StringNumber (page 638), StrokeFinger (page 640), SustainPedal (page 641), TextScript (page 650), and UnaCordaPedal (page 665).

3.2.124 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): LaissezVibrerTieColumn (page 570), and RepeatTieColumn (page 611).

3.2.125 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 (alist, with symbols as keys)
  A list 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:

    annotation (string)
        Annotate a grob for debug purposes.

    note-head (graphical (layout) object)
        A single note head.

This grob interface is used in the following graphical object(s): LaissezVibrerTie
(page 569), and RepeatTie (page 610).

3.2.126 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-extents and extra-spacing-heights 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): NonMusicalPaperColumn
(page 593), NoteColumn (page 595), and PaperColumn (page 600).

3.2.127 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-positioning 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.

The grob interface is used in the following graphical object(s): AccidentalSuggestion (page 478), Arpeggio (page 483), BarNumber (page 489), BassFigureAlignmentPositioning (page 492), CaesuraScript (page 504), CenteredBarNumberLineSpanner (page 507), ClefModifier (page 513), CodaMark (page 515), CombineTextScript (page 517), DoublePercentRepeatCounter (page 533), DynamicLineSpanner (page 538), Episema (page 542), Fingering (page 545), HorizontalBracket (page 557), HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560), JumpScript (page 561), MeasureCounter (page 580), MeasureGrouping (page 582), MeasureSpanner (page 583), MetronomeMark (page 585), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), OttavaBracket (page 599), PercentRepeatCounter (page 603), RehearsalMark (page 607), Script (page 613), SectionLabel (page 615), SegnoMark (page 616), SostenutoPedalLineSpanner (page 624), StanzaNumber (page 633), StringNumber (page 638), StrokeFinger (page 640), SustainPedalLineSpanner (page 642), SystemStartBar (page 644), SystemStartBracket (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), TextScript (page 650), TextSpanner (page 652), TrillPitchAccidental (page 657), TrillPitchGroup
3.2.128 signum-repetitionis-interface
An ancient repeat sign. It is printed with the same infrastructure as bar lines, but it is not a bar line.

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 \textit{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, \textit{glyph-name} represents a processed form of \textit{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 \textit{not} influenced by changes to \textit{Staff}\textperiodcentered StaffSymbol\textperiodcentered 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 \textit{not} influenced by changes to \textit{Staff}\textperiodcentered StaffSymbol\textperiodcentered 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 \textit{not} influenced by changes to \textit{Staff}\textperiodcentered StaffSymbol\textperiodcentered 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 \textit{not} influenced by changes to \textit{Staff}\textperiodcentered StaffSymbol\textperiodcentered 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): SignumRepetitionis (page 618).

### 3.2.129 slur-interface

A slur. Slurs are formatted by trying a number of combinations of left/right end point, and then picking the slur with the lowest demerit score. The combinations are generated by going from the base attachments (i.e., note heads) in the direction in half space increments until we have covered region-size staff spaces. 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:

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 (alist, with symbols as keys)
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-quants (pair of numbers)
If debugging is set, set beam and slur position to a (quantized) position that is as close as possible to this value, and print the demerits for the inspected position in the output.

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 (start, end), where start and end are vertical positions 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:**

annotation (string)
Annotate a grob for debug purposes.

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): PhrasingSlur (page 604), and Slur (page 621).

3.2.130 **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)
A grob similar to this one, on its left. For columns, 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): NonMusicalPaperColumn (page 593), and PaperColumn (page 600).

3.2.131 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): NoteSpacing (page 598), and StaffSpacing (page 632).

3.2.132 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): GraceSpacing (page 553), and SpacingSpanner (page 626).

3.2.133 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 proportionally 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): SpacingSpanner
(page 626).

3.2.134 span-bar-interface
A bar line that is spanned between other bar lines. 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): SpanBar (page 626).
3.2.135 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): BassFigureAlignment (page 492), BassFigureAlignmentPositioning (page 492), BassFigureContinuation (page 494), Beam (page 495), BendAfter (page 497), BendSpanner (page 498), CenteredBarNumber (page 506), CenteredBarNumberLineSpanner (page 507), ChordSquare (page 509), ClusterSpanner (page 514), DurationLine (page 536), DynamicLineSpanner (page 538), DynamicTextSpanner (page 541), Episema (page 542), FingerGlideSpanner (page 543), Glissando (page 552), GraceSpacing (page 553), GridChordName (page 553), Hairpin (page 555), HorizontalBracket (page 557), HorizontalBracketText (page 558), InstrumentName (page 559), KievanLigature (page 568), LedgerLineSpanner (page 570), LigatureBracket (page 573), LyricExtender (page 574), LyricHyphen (page 575), LyricSpace (page 578), MeasureCounter (page 580), MeasureGrouping (page 582), MeasureSpanner (page 583), MensuralLigature (page 585), MultiMeasureRest (page 587), MultiMeasureRestNumber (page 589), MultiMeasureRestScript (page 590), MultiMeasureRestText (page 592), OttavaBracket (page 599), PercentRepeat (page 602), PercentRepeatCounter (page 603), PhrasingSlur (page 604), PianoPedalBracket (page 606), Slur (page 621), SostenutoPedalLineSpanner (page 624), SpacingSpanner (page 626), StaffGrouper (page 630), StaffHighlight (page 631), StaffSymbol (page 632), SustainPedalLineSpanner (page 642), System (page 643), SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), SystemStartSquare (page 647), TextSpanner (page 652), Tie (page 653), TieColumn (page 655), TrillSpanner (page 661), TupletBracket (page 662), TupletNumber (page 664), UnaCordaPedalLineSpanner (page 666), VaticanaLigature (page 667), VerticalAlignment (page 668), VerticalAxisGroup (page 668), VoiceFollower (page 670),
VoltaBracket (page 671), VoltaBracketSpanner (page 672), and VowelTransition (page 673).

In addition, this interface is supported conditionally by the following objects depending on their class: BalloonText (page 484), ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), and Parentheses (page 601).

3.2.136 staff-grouper-interface
A grob that collects staves together.

User settable properties:

staff-staff-spacing (alist, with symbols as keys)
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 (alist, with symbols as keys)
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): StaffGrouper (page 630).

3.2.137 staff-highlight-interface
A colored span to highlight a music passage.

User settable properties:

bound-prefatory-paddings (pair of numbers)
For a highlight, the amount of padding to insert at a bound from a prefatory item that is not a bar line.

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:

- **columns** (array of grobs)
  An array of grobs, typically containing PaperColumn or NoteColumn objects.

- **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): StaffHighlight (page 631).

### 3.2.138 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): StaffSpacing (page 632).

### 3.2.139 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).

widened-extent (pair of numbers)
   The vertical extent that a bar line on a certain staff symbol should have. If the staff
   symbol is small (e.g., has just one line, as in a RhythmicStaff, this is wider than the
   staff symbol’s Y extent.

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): StaffSymbol (page 632).

3.2.140 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): AmbitusNoteHead  
(page 482), Arpeggio (page 483), Beam (page 495), Clef (page 510), CueClef (page 521),
CueEndClef (page 524), Custos (page 527), Dots (page 531), KeyCancellation (page 563),
KeySignature (page 565), MultiMeasureRest (page 587), NoteHead (page 596), Rest  
(page 611), TabNoteHead (page 648), and TrillPitchHead (page 659).

3.2.141 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): StanzaNumber (page 633).

3.2.142 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 (alist, with symbols as keys)
A list 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 (each unit represents half a staff-space).

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.

french-beaming-stem-adjustment (dimension, in staff space)
   Stem will be shortened by this amount of space in case of French beaming style.

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): Stem (page 634).

3.2.143 stem-tremolo-interface

A beam slashing a stem to indicate a tremolo. The property shape can be beam-like or rectangle.
Chapter 3: Backend

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): StemTremolo (page 637).

3.2.144 sticky-grob-interface

A grob that is attached to another grob. Grobs type having this interface can be either items
or spanners, depending on the class of their host. Sticky spanners implicitly take their bounds
from the host.

Internal properties:

sticky-host (graphical (layout) object)
  The grob that a sticky grob attaches to.

This grob interface is used in the following graphical object(s): BalloonText (page 484),
ControlPoint (page 519), ControlPolygon (page 520), Footnote (page 548), and
Parentheses (page 601).

3.2.145 string-number-interface

A string number instruction.

This grob interface is used in the following graphical object(s): StringNumber (page 638).

3.2.146 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): StrokeFinger (page 640).
3.2.147 system-interface
This is the top-level object: Each object in a score ultimately has a System object as its X and Y parent.

The system-interface implies the spanner-interface (page 746).

User settable properties:

labels (list)
List of labels (symbols) placed on a column.

page-number (number)
Page number on which this system ends up.

rank-on-page (number)
0-based index of the system on a page.

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): System (page 643).

3.2.148 system-start-delimiter-interface
The brace, bracket or bar in front of the system. The following values for style are recognized:

brace
A ‘piano style’ brace normally used for an instrument that uses two staves. The default style for GrandStaff. SystemStartBrace uses this style.

bracket
A thick bracket, normally used to group similar instruments in a score. Default for StaffGroup. SystemStartBracket 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): SystemStartBar (page 644), SystemStartBrace (page 645), SystemStartBracket (page 646), and SystemStartSquare (page 647).

3.2.149 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): InstrumentName (page 559).

3.2.150 tab-note-head-interface
A note head in tablature.
User settable properties:

- **details** (alist, with symbols as keys)
  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): TabNoteHead (page 648).

### 3.2.151 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** (association list (list of pairs))
  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): BalloonText (page 484), BarNumber (page 489), BassFigure (page 491), BendSpanner (page 498), BreathingSign (page 502), CenteredBarNumber (page 506), ChordName (page 508), ClefModifier (page 513), CodaMark (page 515), CombineTextScript (page 517), ControlPoint (page 519), ControlPolygon (page 520), Divisio (page 528), DoublePercentRepeatCounter (page 533), DynamicText (page 539), DynamicTextSpanner (page 541), Fingering (page 545), Footnote (page 548), GridChordName (page 553), HorizontalBracketText (page 558), InstrumentName (page 559), InstrumentSwitch (page 560), JumpScript (page 561), LyricRepeatCount (page 576), LyricText (page 579), MeasureCounter (page 580), MeasureSpanner (page 583), MetronomeMark (page 585), MultiMeasureRestNumber (page 589), MultiMeasureRestText (page 592), NoteName (page 597), OttavaBracket (page 599), PercentRepeatCounter
3.2.152 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): BendSpanner (page 498), CombineTextScript (page 517), Fingering (page 545), StringNumber (page 638), StrokeFinger (page 640), and TextScript (page 650).

3.2.153 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): TieColumn (page 655).
3.2.154 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:

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 (alist, with symbols as keys)
A list 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).

Internal properties:

annotation (string)
Annotate a grob for debug purposes.

This grob interface is used in the following graphical object(s): LaissezVibrerTie (page 569), RepeatTie (page 610), and Tie (page 653).

3.2.155 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 value ‘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): `TimeSignature` (page 655).

### 3.2.156 trill-pitch-accidental-interface

An accidental for trill pitch.

This grob interface is used in the following graphical object(s): `TrillPitchAccidental` (page 657).

### 3.2.157 trill-spanner-interface

A trill spanner.

This grob interface is used in the following graphical object(s): `TrillSpanner` (page 661).

### 3.2.158 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.

- **dashed-edge** *(boolean)*
  If set, the bracket edges are dashed like the rest of the bracket.

- **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 (start . end), where start and end are vertical positions 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).

tuplet-slur (boolean)
   Draw a slur instead of a bracket for tuplets.

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): LigatureBracket (page 573), and TupletBracket (page 662).

3.2.159 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): TupletNumber (page 664).

3.2.160 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): Beam (page 495), DurationLine (page 536), and Glissando (page 552).

3.2.161 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 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): NoteHead (page 596), and VaticanaLigature (page 667).

3.2.162 volta-bracket-interface
Volta bracket with number.

User settable properties:

dashed-edge (boolean)
  If set, the bracket edges are dashed like the rest of the bracket.

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): VoltaBracket (page 671).

3.2.163 volta-interface
A volta repeat.

This grob interface is used in the following graphical object(s): VoltaBracket (page 671), and VoltaBracketSpanner (page 672).
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 (association list (list of pairs))
   List of (pitch, accidental) pairs for key signature.

alteration-glyph-name-alist (association list (list of pairs))
   An alist of key-string pairs.

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 and it is larger than this number, make a kneed beam.

automatically-numbered (boolean)
   If set, footnotes are 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.

bend-me (boolean)
Decide whether this grob is bent.

between-cols (pair)
Where to attach a loose column to.

bound-details (alist, with symbols as keys)
An alist of properties for determining attachments of spanners to edges.

bound-padding (number)
The amount of padding to insert around spanner bounds.

bound-prefatory-paddings (pair of numbers)
For a highlight, the amount of padding to insert at a bound from a prefatory item that is not a bar line.
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:

   ```ly
   \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 (alist, with symbols as keys)
   An alist of parent-alignments that should be used for clef modifiers with various clefs

collapse-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 collision apply only to the voice in which the beam was created?

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

color (color)
   The color of this grob.

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.

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.

dashed-edge (boolean)
   If set, the bracket edges are dashed like the rest of the bracket.
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 (alist, with symbols as keys)
  A list 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)
  A 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).

endpoint-alignments (pair of numbers)
  A pair of numbers representing the alignments of an object’s endpoints. E.g., the ends of a
  hairpin relative to NoteColumn grobs.

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).

extroversion (number)
   For polygons, how the thickness of the line is spread on each side of the exact polygon with
ideal zero thickness. If this is 0, the middle of line is on the polygon. If 1, the line sticks out
of the polygon. If -1, the outer side of the line is exactly on the polygon. Other numeric
values are interpolated.

fa-merge-direction (direction)
   If two ‘fa’ shape note heads get merged that are both listed in the fa-styles property but
have different stem directions, enforce this note head direction for display.

filled (boolean)
   Whether an object is filled with ink.

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 (Em-
mentaler), 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 (alist, with symbols as keys)
    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-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.

• **handedness** – Print the fret-diagram left- or right-handed. -1, LEFT for left ; 1, RIGHT for right. Default RIGHT.

• **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 arabic, roman-ij-lower, roman-ij-upper, roman-lower, roman-upper, arabic and custom. In the last 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 \( \text{thickness} \times (1 + \text{string-thickness-factor}^k) \). 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-left (string)
The glyph value to use at the end of the line when the line is broken. #f indicates that no
glyph should be visible; otherwise the value must be a string.

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-right (string)
The glyph value to use at the beginning of the line when the line is broken. #f indicates
that no glyph should be visible; otherwise the value must be a string.

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 de-
fault staff-line thickness (i.e., the visual output is not influenced by changes to
Staff.StaffSymbol.thickness).

harp-pedal-details (alist, with symbols as keys)
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. De-
fault value 0.15.
• circle-y-padding – Padding in Y direction of the ellipse around circled pedals. De-
fault 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 NoteColumns 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-quants (pair of numbers)
   If debugging is set, set beam and slur position to a (quantized) position that is as close as possible to this value, and print the demerits for the inspected position in the output.

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 knedd?

knee-spacing-correction (number)
   Factor for the optical correction amount for knedd 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 knedd 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 (alist, with symbols as keys)
An alist of properties for determining attachments of spanners to edges.

left-number-text (markup)
For a measure counter, this is the formatted measure count. When the measure counter extends over several measures (like with compressed multi-measure rests), it is the text on the left side of the dash.

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 (each unit represents half a staff-space).

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 (alist, with symbols as keys)
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.

main-extent (pair of numbers)
The horizontal extent of a NoteColumn grob without taking suspended NoteHead grobs into account (i.e., NoteHeads forced into the unnatural direction of the Stem because of a chromatic clash).
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-division (number list)
  A list representing what fraction of the measure length each chord name takes in a chord square. The list is made of exact numbers between 0 and 1, which should add up to 1. Example: a measure c2 g4 g4 results in '(1/2 1/4 1/4).

measure-division-chord-placement-alist (association list (list of pairs))
  An alist mapping measure divisions (see the measure-division property) to lists of coordinates (number pairs) applied to the chord names of a chord square. Coordinates are normalized between -1 and 1 within the square.

measure-division-lines-alist (association list (list of pairs))
  An alist mapping measure divisions (see the measure-division property) to lists of lines to draw in the square, given as 4-element lists: (x-start y-start x-end y-end).

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-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 (alist, with symbols as keys)
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 (alist, with symbols as keys)
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 (alist, with symbols as keys)
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-range-separator** (markup)
For a measure counter extending over several measures (like with compressed multi-measure rests), this is the separator between the two printed numbers.

**number-type** (symbol)
Numbering style. Choices include arabic, roman-ij-lower, roman-ij-upper, roman-lower, and roman-upper.

**output-attributes** (association list (list of pairs))
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 (\texttt{<g>}) containing all of the stencils that comprise a given grob. For example,

\begin{verbatim}
'((id . 123) (class . foo) (data-whatever . "bar"))
\end{verbatim}

produces

\begin{verbatim}
<g id="123" class="foo" data-whatever="bar"> ... </g>
\end{verbatim}

In the Postscript backend, where there is no way to group items, the setting of the output-attributes property has no effect.

**outside-staff-horizontal-padding** (number)
By default, an outside-staff-object can be placed so that it is 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** (association list (list of pairs))
An alist of padding pairs for key signatures (and key cancellations). Each alist entry has the form

\begin{verbatim}
((left-glyph-name . right-glyph-name) . dist)
\end{verbatim}

specifying the padding \texttt{dist} between two adjacent key signature elements. If there is no entry in the alist for a given pair, the padding value given by the padding property of the KeySignature (or KeyCancellation) grob is used instead.
A special feature is the handling of adjacent naturals (to be more precise, the handling of glyph accidentals.natural): If there is no ‘natural-natural’ entry in padding-pairs explicitly overriding it, LilyPond adds some extra padding (in addition to the grob’s padding value) to avoid collisions.

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-number (number)
Page number on which this system ends up.

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.

parenthesis-id (symbol)
When parenthesized grobs created in the same time step have this property, there is one set of parentheses for each group of grobs having the same value.

parenthesized (boolean)
Parenthesize this grob.

positions (pair of numbers)
Pair of staff coordinates (start . end), where start and end are vertical positions 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.
rank-on-page (number)
  0-based index of the system on a page.

different (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 (association list (list of pairs))
  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.

different-location (rhythmic location)
  Where (bar number, measure position) in the score.

right-bound-info (alist, with symbols as keys)
  An alist of properties for determining attachments of spanners to edges.

right-number-text (markup)
  When the measure counter extends over several measures (like with compressed multi-measure rests), this is the text on the right side of the dash. Usually unset.

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.

show-control-points (boolean)
For grobs printing Bézier curves, setting this property to true causes the control points and control polygon to be drawn on the page for ease of tweaking.

show-horizontal-skylines (boolean)
If true, print this grob’s horizontal skylines. This is meant for debugging purposes.

show-vertical-skylines (boolean)
If true, print this grob’s vertical skylines. This is meant for debugging purposes.

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.

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-extents and extra-spacing-heights 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 (alist, with symbols as keys)
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 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)

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 (alist, with symbols as keys)
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).

staff-group-staff-spacing (alist, with symbols as keys)
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-alignment-X (number)
   How to align an annotation horizontally.

text-alignment-Y (number)
   How to align an annotation vertically.

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.
tuplet-slur (boolean)
   Draw a slur instead of a bracket for tuplets.

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-logs 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 be-
   yond 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.

widened-extent (pair of numbers)
   The vertical extent that a bar line on a certain staff symbol should have. If the staff symbol
   is small (e.g., has just one line, as in a RhythmicStaff, this is wider than the staff symbol’s
   Y extent.

width (dimension, in staff space)
   The width of a grob measured in staff space.

woodwind-diagram-details (alist, with symbols as keys)
   An alist of detailed grob properties for woodwind diagrams. Each alist entry consists of a
   (property . value) pair. The properties which can be included in woodwind-diagram-
   details include the following:
   • fill-angle – Rotation angle of a partially filled key from horizontal. Default value 0.
   • text-trill-circled – In non-graphical mode, for keys shown as text, indicate a trill
     by circling the text if true, or by shading the text if false. Default value #t.

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-attachment (number)
   Horizontal attachment of a line on a frame, typically between -1 (left) and 1 (right).
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 \((\text{left} . \text{right})\), where both \text{left} and \text{right} are in staff-space units of the current staff.

Y-attachment (number)
   Vertical attachment of a line on a frame, typically between -1 (down) and 1 (up).

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 spanner 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 (association list (list of pairs))
   An alist with \((\text{notename} . \text{groblist})\) 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 \texttt{VerticalAxisGroup} to cache the Y-extents 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.

annotation (string)
   Annotate a grob for debug purposes.

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.

bezier (graphical (layout) object)
   A pointer to a Bézier curve, for use by control points and polygons.

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.

break-alignment (graphical (layout) object)
   The BreakAlignment (page 501), in a NonMusicalPaperColumn (page 593).

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?

chord-names (array of grobs)
   Array of chord names.

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 diminished?

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.

fa-styles (symbol list)
List of note head styles that identify ‘fa’ shape note heads.

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 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.

default-beam-width (dimension, in staff space)
Stem will be shortened by this amount of space in case of default beam style.

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?

index (non-negative, exact integer)
For some grobs in a group, this is a number associated with the grob.

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 VerticalAxisGroups. 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)
   A grob similar to this one, on its left. For columns, 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 VerticalAxisGroups. 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 (alist, with symbols as keys)
   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 (association list (list of pairs))
   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.

sticky-host (graphical (layout) object)
The grob that a sticky grob attaches to.

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

**add-bar-glyph-print-procedure**  
*glyph proc*  
Specify the single glyph *glyph* that calls print procedure *proc*. The procedure *proc* has to be defined in the form `make-...-bar-line grob extent` even if the extent is not used within the routine.

**ly:add-context-mod**  
*contextmods modification*  
Adds the given context *modification* to the list *contextmods* of context modifications.

**add-grace-property**  
*context-name grob sym val*  
Set *sym* = *val* for *grob* in *context-name*.

**ly:add-interface**  
*iface desc props*  
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*  
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.

**add-music-fonts**  
*node family name brace design-size-alist factor*  
Set up music fonts.  
Arguments:  
- *node* is the font tree to modify.  
- *family* is the family name of the music font.  
- *name* is the basename for the music font. *name*-<designsize>.otf should be the music font.  
- *brace* is the basename for the brace font. *brace*-brace.otf should have piano braces.  
- *design-size-alist* is a list of `(rounded . designsize)`. *rounded* is a suffix for font filenames, while *designsize* should be the actual design size. The latter is used for text fonts loaded through pango/fontconfig.  
- *factor* is a size factor relative to the default size that is being used. This is used to select the proper design size for the text fonts.

**add-new-clef**  
*clef-name clef-glyph clef-position transposition c0-position*  
Append the entries for a clef symbol to supported clefs and c0-pitch-alist.

**ly:add-option**  
*sym val internal description*  
Add a program option *sym*. *val* is the default value and *description* is a string description.

**add-simple-time-signature-style**  
*style proc*  
Specify the procedure *proc* returning markup for a time signature style *style*. The procedure is called with one argument, the pair `(numerator , denominator)`.

**add-stroke-glyph**  
*stencil grob dir stroke-style flag-style*  
Load and add a stroke (represented by a glyph in the font) to the given flag stencil.

**add-stroke-straight**  
*stencil grob dir log stroke-style offset length thickness stroke-thickness*  
Add the stroke for acciaccatura to the given flag stencil. The stroke starts for up-flags at `'(upper-end-of-flag + (0,length/2))` and ends at `'(0, vertical-center-of-flag-end) - (flag-x-width/2, flag-x-width + flag-thickness)`.

For down-stems the y coordinates are simply mirrored.
alist->hash-table lst
Convert alist lst to a table.

**Warning:** The resulting hash table is hashed by identity. This actually corresponds to the alist->hashq-table function of Guile's (ice-9 hash-table) module, not alist->hash-table.

ly:all-grob-interfaces
Return the hash table with all grob interface descriptions.

ly:all-options
Get all option settings in an alist.

ly:all-output-backend-commands
Return the list of extra output backend commands that are used internally in file lily/stencil-interpret.cc.

ly:all-stencil-commands
Return the list of stencil commands that can be defined in the output modules (in files output-*-scm).

ly:all-stencil-expressions
Return all symbols recognized as stencil expressions.

allow-volta-hook bar-glyph
Allow the volta bracket hook being drawn over bar line bar-glyph.

alterations-in-key pitch-list
Count number of sharps minus number of flats.

ly:angle x y
Calculate 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.

angle-0-2pi angle
Take angle (in radians) and map it between 0 and 2pi.

angle-0-360 angle
Take angle (in degrees) and map it between 0 and 360 degrees.

arrow-stencil x y thick staff-space grob
Return a right-pointing, filled arrow-head, where x determines the basic horizontal position and y determines the basic vertical position. Both values are adjusted using staff-space, which is StaffSymbol’s staff space. thick is the used line thickness.

arrow-stencil-maker start? end?
Return a function drawing a line from current point to destination, with optional arrows of max-size on start and end controlled by start? and end?.

ly:assoc-get key alist default-value strict-checking
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.

assoc-get . . [...] - LilyPond procedure: ly:assoc-get (SCM key, SCM alist, SCM default_value, SCM strict_checking)
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.
Chapter 4: Scheme functions

at-bar-line-substitute-caesura-type \textit{substitute-type} \hspace{1cm} \textbf{[Function]}
At a bar line, create the caesura using \textit{substitute-type} rather than the value of \textit{caesuraType}.

\texttt{ly:axis-group-interface::add-element \textit{grob} \textit{grob-element}} \hspace{1cm} \textbf{[Function]}
Add \textit{grob-element} to the axis group \textit{grob}. In particular, \textit{grob} becomes parent to \textit{grob-element} on all axes supported by \textit{grob}, unless the parents are already set.

\texttt{ly:bar-line::calc-anchor \textit{grob}} \hspace{1cm} \textbf{[Function]}
Calculate the anchor position of a bar line. The anchor is used for the correct placement of bar numbers, etc.

\texttt{bar-line::calc-break-visibility \textit{grob}} \hspace{1cm} \textbf{[Function]}
Calculate the visibility of a bar line at line breaks.

\texttt{bar-line::calc-glyph-name \textit{grob}} \hspace{1cm} \textbf{[Function]}
Return the name of the bar line glyph printed by \textit{grob} for the actual break direction.

\texttt{bar-line::calc-glyph-name-for-direction glyphs \textit{dir}} \hspace{1cm} \textbf{[Function]}
Find the glyph name for a bar line. \textit{glyphs} is the list of bar-line types to consider in order. Each must have been defined with define-bar-line. \textit{dir} is the break direction to consider: LEFT = end of line, CENTER = middle of line, RIGHT = start of line.

\texttt{bar-line::compound-bar-line \textit{grob} \textit{bar-glyph} \textit{extent}} \hspace{1cm} \textbf{[Function]}
Build the bar line stencil.

\texttt{bar-line::draw-filled-box x-ext y-ext \textit{thickness} \textit{extent} \textit{grob}} \hspace{1cm} \textbf{[Function]}
Return a straight bar line created by \texttt{ly:round-filled-box} looking at \textit{x-ext}, \textit{y-ext}, \textit{thickness}. The blot is calculated by \texttt{bar-line::calc-blot}, which needs \textit{extent} and \textit{grob}. \textit{y-ext} is not necessarily of same value as \textit{extent}.

\texttt{ly:bar-line::print \textit{grob}} \hspace{1cm} \textbf{[Function]}
The print routine for bar lines.

\texttt{bar-line::widen-bar-extent-on-span \textit{grob} \textit{extent}} \hspace{1cm} \textbf{[Function]}
Widen the bar line \textit{extent} towards span bars adjacent to \textit{grob} \textit{grob}.

\texttt{base-length \textit{time-signature} \textit{time-signature-settings}} \hspace{1cm} \textbf{[Function]}
Get \texttt{baseMoment} rational value for \textit{time-signature} from \textit{time-signature-settings}.

\texttt{ly:basic-progress \textit{str} \textit{rest}} \hspace{1cm} \textbf{[Function]}
A Scheme callable function to issue a basic progress message \textit{str}. The message is formatted with \texttt{format}; \textit{rest} holds the formatting arguments (if any).

\texttt{beam-exceptions \textit{time-signature} \textit{time-signature-settings}} \hspace{1cm} \textbf{[Function]}
Get beamExceptions value for \textit{time-signature} from \textit{time-signature-settings}.

\texttt{beat-structure \textit{base-length} \textit{time-signature} \textit{time-signature-settings}} \hspace{1cm} \textbf{[Function]}
Get beatStructure value in \textit{base-length} units for \textit{time-signature} from \textit{time-signature-settings}.

\texttt{bend::arrow-head-stencil \textit{thickness} x-y-coords \textit{height} \textit{width} \textit{dir}} \hspace{1cm} \textbf{[Function]}
Return an arrow head stencil, calculated from the given dimensions \textit{height} and \textit{width}, and translated to \textit{x-y-coords}, the end of the bend-spanners (curved) line.
bend::calc-bend-x-begin bend-spanner bounding-noteheads factor [Function]

calculate the starting values in x direction of the bend. After a line break, the values from the right bound are taken minus 1.5 staff spaces. For bends-down or if grob property 'style equals to 'pre-bend, 'hold or 'pre-bend-hold, interval-center is applied the topmost note head of the starting note heads. In any other case the right edge of the starting note head is used. The value of BendSpanner.details.horizontal-left-padding is added, which may be changed by an appropriate override. Returns a list of the same length as the amount of bend-starting note heads.

bend::calc-bend-x-end bend-spanner top-left-tab-nhd [Function]

calculate the ending x coordinate of bend-spanner. At the line end, take the items of BreakAlignGroup into account and a little bit of padding. Ends an unbroken spanner or the last of a broken one in the middle of the topmost note head of its bounding note column.

bend::target-cautionary spanner [Function]

set 'display-cautionary of all relevant note heads of spanners right bound to true. As a result they appear parenthesized. This procedure is the default value of 'before-line-breaking.

bend::text-string spanner [Function]

take a spanner grob and calculate a list with the quarter tone diffs between the pitches of starting and ending bound. Because bending to different amounts is very unlikely, only the first element of this list is returned as a string.

bend-spanner::print grob [Function]

return the final stencil. A line and curve, an arrow head and a text representing the amount a string is bent.

ly:bezier-extent control-points axis [Function]

compute the extent of the Bézier curve defined by control-points along axis.

ly:bezier-extract control-points t-min t-max [Function]

return a sub-curve of the Bézier curve defined by control-points. The sub-curve is delimited by the curve points indexed by t-min and t-max (between 0 and 1, 0 = first control point, 1 = last control point). A sub-curve of a Bézier curve is in turn a Bézier curve.

ly:book? x [Function]

is x a smob of class Book?


add score to book-smob score list.


book-first-page layout props [Function]

return the 'first-page-number of the entire book.


return header in book.
   Return paper in  book.

   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).

   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
   Return scores in  book.

   Set the book header.

box-grob-stencil  grob
   Make a box of exactly the extents of the grob. The box precisely encloses the contents.

box-stencil  stencil thickness padding
   Add a box around  stencil, producing a new stencil.

ly:bp num
   num  bigpoints (1/72th inch).

ly:bracket a iv t p
   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.

bracketify-stencil still axis thick protrusion padding
   Add brackets around  still, producing a new stencil.

break-alignable-interface::self-alignment-of-anchor g
   Return a value for  g’s  self-alignment- X  that will place  g  on the same side of the reference
   point defined by a break-aligned item such as a Clef.

break-alignable-interface::self-alignment-opposite-of-anchor g
   Return a value for  g’s  self-alignment- X  that will place  g  on the opposite side of the reference
   point defined by a break-aligned item such as a Clef.

ly:break-alignment-interface::find-nonempty-break-align-group
   Find the BreakAlignGroup with the given break-align-symbol in this BreakAlignment. Return
   #f  if there is no such group. Also return #f  if the group has empty X-extent, which
   can happen if it contains only omitted items.

break-alignment-list  end-of-line middle begin-of-line
   Return a callback that calculates a value based on a grob’s break direction.

ly:broadcast  disp ev
   Send the stream event  ev  to the dispatcher  disp.

caesura-script-interface::before-line-breaking script
   Callback for CaesuraScript grob. Eliminate scripts aligned to bar lines if they might collide
   with a span bar. Some types of bar lines have visible span bars and some don’t. For consistent
   notation, we don’t check whether particular SpanBar grobs are actually visible, just that they
   exist.
caesura-to-bar-line-or-divisio context caesura-type observations  [Function]
   caesuraTypeTransform callback to print articulated caesurae as chant breath marks using
   the infrastructure for modern bar lines when possible.

caesura-to-divisio context caesura-type observations  [Function]
   caesuraTypeTransform callback to print articulated caesurae as chant breath marks.

ly:cairo-output-stencil basename stencil paper formats  [Function]
   dump a single stencil through the Cairo backend

ly:cairo-output-stencils basename stencils header paper formats  [Function]
   dump book through cairo backend

calc-harmonic-pitch pitch music  [Function]
   Calculate the harmonic pitches in music given pitch as the non-harmonic pitch.

ly:camel-case->lisp-identifier name-sym  [Function]
   Convert FooBar_Bla to foo-bar-bla style symbol.

centered-spanner-interface::calc-x-offset grob  [Function]
   Compute the shift from this spanner's reference point to a point centered between two non-
   musical columns, according to the spacing-pair property. This also takes self-alignment-X
   into account. The default for spacing-pair is '(break-alignment . break-alignment).

centered-stencil stencil  [Function]
   Center stencil stencil in both the x and y directions.

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.

chain-assoc-get . . [ ]  [Function]
   - LilyPond procedure: ly:chain-assoc-get (SCM key, SCM achain, SCM default_value, SCM
     strict_checking)
   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.

change-pitches music converter  [Function]
   Recurse through music, applying converter to pitches. converter is typically a transposer or
   an inverter (see file scm/modal-transforms.scm), but may be user-defined. The converter
   function must take a single pitch as its argument and return a new pitch. These are LilyPond
   Scheme pitches, e.g., (ly:make-pitch 0 2 0).

check-context-path path [location]  [Function]
   Check a context property path specification path, a symbol list (or a single symbol), for
   validity and possibly complete it. Returns the completed specification, or #f when rising an
   error (using optionally location).

ly:check-expected-warnings  [Function]
   Check whether all expected warnings have really been triggered.
Chapter 4: Scheme functions

check-grob-path path rest...
[Function]
Check a grob path specification path, a symbol list (or a single symbol), for validity and possibly complete it. Returns the completed specification, or #f if invalid, optionally using location for an error message. If an optional keyword argument #:start start is given, the parsing starts at the given index in the sequence ‘Context.Grob.property.sub-property...’, with the default of ‘0’ implying the full path.

If there is no valid first element of path fitting at the given path location, an optionally given #:default default is used as the respective element instead without checking it for validity at this position.

The resulting path after possibly prepending default can be constrained in length by optional arguments #:min min and #:max max, defaulting to ‘1’ and unlimited, respectively.

check-music-path path rest...
[Function]
Check a music property path specification path, a symbol list (or a single symbol), for validity and possibly complete it. Returns the completed specification, or #f when rising an error (using optionally location).

choir-name->german-markup B-instead-of-Bb
[Function]
Return pitch markup for PITCH, using german note names. If B-instead-of-Bb is set to #t real german names are returned. Otherwise semi-german names (with Bb and below keeping the british names)

chiro-name->italian-markup french?
[Function]
Return pitch markup for pitch, using Italian/French note names. If french? is set to #t, french ‘ré’ is returned for pitch D instead of ‘re’.

circle-stencil stencil thickness padding
[Function]
Add a circle around stencil, producing a new stencil.

clef-transposition-markup oct style
[Function]
The transposition sign formatting function. oct is supposed to be a string holding the transposition number, style determines the way the transposition number is displayed.

ly:cm num
num cm.
[Function]

Book music handler.
[Function]

collect-bookpart-for-book book-part
Top-level book-part handler.
[Function]

collect-music-aux score-handler music
Pass music to score-handler, with preprocessing for page layout instructions.
[Function]

collect-music-for-book music
Top-level music handler.
[Function]

ly:command-line-code
The Scheme code specified on the command line with option `-e`.
[Function]

ly:command-line-options
The Scheme options specified on the command line with option `-d`.
[Function]

comparator-from-key key cmp
Return a comparator function that applies key to the two elements and compares the results using cmp. Especially useful for sorting.
ly:connect-dispatchers to from
Make the dispatcher to listen to events from from.

construct-chord-elements root duration modifications
Build a chord on root using modifiers in modifications. NoteEvents have duration duration.
Notes: Natural 11 is left from chord if not explicitly specified.
Entry point for the parser.

ly:context? x
Is x a smob of class Context?

ly:context-current-moment context
Return the current moment of context.

ly:context-def? x
Is x a smob of class Context_def?

ly:context-def-lookup def sym val
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
Return the result of applying the context-mod mod to the context definition def. Does not change def.

ly:context-event-source context
Return event-source of context context.

ly:context-events-below context
Return a stream-distributor that distributes all events from context and all its subcontexts.

ly:context-find context name
Find a parent of context that has name or alias name. Return #f if not found.

ly:context-grob-definition context name
Return the definition of name (a symbol) within context as an alist.

ly:context-id context
Return the ID string of context, i.e., for \context Voice = "one" ... return the string one.

ly:context-matched-pop-property context grob cell
This undoes a particular \override, \once \override or \once \revert when given the specific alist pair to undo.

ly:context-mod? x
Is x a smob of class Context_mod?

ly:context-mod-apply! context mod
Apply the context modification mod to context.

ly:context-name context
Return the name of context, i.e., for \context Voice = "one" ... return the symbol Voice.

ly:context-output-def context
Return the output definition of context.
ly:context-parent context
Return the parent of context, #f if none.

ly:context-property context sym def
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 def
Return the context above context where name is defined, or def (defaulting to '() ) if no such context is found.

ly:context-pushpop-property context grob eltprop val
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
Set value of property name in context context to val.

copy-repeat-chord original-chord repeat-chord duration event-types
Copy all events in event-types (be sure to include rhythmic-events) from original-chord over to repeat-chord with their articulations filtered as well. Any duration is replaced with the specified duration.

count-list lst
Given lst as (E1 E2 .. ), return ((E1 . 1) (E2 . 2) ... ).

create-glyph-flag flag-style dir-modifier grob
Create a flag stencil by looking up the glyph from the font.

cross-staff-connect stem
Set cross-staff property of the stem to this function to connect it to other stems automatically.

cue-substitute quote-music
Must happen after quote-substitute.

cyclic-base-value value cycle
Take value (for example, an angle) and modulo-maps it between 0 and base cycle.

ly:debug str rest
A Scheme callable function to issue a debug message str. The message is formatted with format; rest holds the formatting arguments (if any).

default-flag grob
Create a flag stencil for the stem. Its style is derived from the 'style Flag property. By default, Lilypond uses a C++ function (which is slightly faster) to do exactly the same as this function. However, if one wants to modify the default flags, this function can be used to obtain the default flag stencil, which can then be modified at will. The correct way to do this is:

\override Flag #'stencil = #default-flag
\override Flag #'style = #'mensural
ly:default-scale

Get the global default scale.

define-bar-line bar-glyph eol-glyph bol-glyph span-glyph

Define a bar glyph bar-glyph and its substitutes at the end of a line (eol-glyph), at the beginning of a line (bol-glyph) and as a span bar (span-glyph). The substitute glyphs may be either strings or booleans: #t calls for the same value as bar-glyph and #f calls for no glyph.

define-event-class class parent

Defines a new event class derived from parent, a previously defined event class.

define-event-function ...

Like define-music-function, but the return value must be a post-event.

define-fonts paper define-font define-pango-pf

Return a string of all fonts used in paper, invoking the functions define-font and define-pango-pf for producing the actual font definition.

define-markup-command ...

Define a markup function. Syntax:

```
(define-markup-command (command layout props arg1 arg2 ...) (type1? type2? ...) [ #:properties ((property1 default1) (property2 default2) ...) ] [ #:category category ] [ #:as-string expression ] [ "doc-string" ] command-body)
```

This macro defines the markup function command-markup. When this function is applied as

```
(command-markup layout props arg1 arg2 ...)
```

it executes command-body, a sequence of S-expression similar to the body of a define form. The body should return a stencil.

type1?, type2?, etc., are type predicates for the arguments arg1, arg2, etc. doc-string is an optional description of the command; this can be retrieved using procedure-documentation on command-markup, and is used for built-in markup commands to generate the documentation.

Moreover, this macro defines a helper function make-command-markup, which can be applied as

```
(make-command-markup arg1 arg2 ...)
```

(without layout and props arguments). This yields a markup. Interpreting it, using (interpret-markup markup layout props), invokes command-markup as above.

The specified properties are available as let-bound variables in the command body, using the respective default value as fallback in case the property is not found in props, or #f if no default was given. props itself is left unchanged: if you want defaults specified in that manner passed down into other markup functions, you need to adjust props yourself.

If the as-string named argument is given, it should be an expression, which is evaluated by markup->string when lossily converting markups to strings. The expression can use all variables available in the main body, namely layout, props, the arguments, and the properties. However, in many cases layout will be #f because such an output definition is
not available (such as for MIDI output). This case must be accounted for. The expression can recursively call \texttt{markup->string}, passing it \texttt{#:layout layout #:props props}.

The autogenerated documentation makes use of some optional specifications that are otherwise ignored:

- \textit{category} is either a symbol or a symbol list specifying the categories for this markup command in the docs.
- As an element of the ‘properties’ list, you may directly use \texttt{command-markup} instead of a \texttt{(property default)} to indicate that this markup command is called by the newly defined command, adding its properties to the documented properties of the new command.

There is no protection against circular definitions.

Some object properties are attached to the resulting \texttt{command-markup} function according to the parameters of the definition: \texttt{markup-command-signature}, \texttt{markup-function-category}, \texttt{markup-function-properties}.

\texttt{define-markup-list-command} ... \hspace{1cm} \textbf{[Macro]}
Same as \texttt{define-markup-command}, but defines a command that, when interpreted, returns a list of stencils instead of a single one.

Markup list commands are recognizable programmatically by having the \texttt{markup-list-function?} object property to \texttt{#t}.

\texttt{define-music-function} ... \hspace{1cm} \textbf{[Macro]}
Define and return a music function. Syntax:

\begin{verbatim}
(define-music-function (arg1 arg2 ...) (type1? type2? ...)
  function-body)
\end{verbatim}

type1?, type2?, etc., can take one of the forms \texttt{predicate?} for mandatory arguments satisfying the predicate, \texttt{(predicate?)} for optional parameters of that type defaulting to \texttt{#f}, \texttt{(predicate? value)} for optional parameters with a specified default value (evaluated at definition time). An optional parameter can be omitted in a call only when it cannot get confused with a following parameter of different type.

A music function must return a music expression.

\texttt{define-scheme-function} ... \hspace{1cm} \textbf{[Macro]}
Like \texttt{define-music-function}, but the return type is not restricted to music.

\texttt{define-syntax-function} ... \hspace{1cm} \textbf{[Macro]}
Helper macro for \texttt{ly:make-music-function}. Syntax:

\begin{verbatim}
(define-syntax-function result-type? (arg1 arg2 ...) (type1? type2? ...)
  function-body)
\end{verbatim}

See \texttt{define-music-function} for information on type predicates. result-type? can specify a default in the same manner as predicates, to be used in case of a type error in arguments or result.

\texttt{define-tag-group} tags \hspace{1cm} \textbf{[Function]}
Define a tag group consisting of the given tags, a list of symbols. Returns \texttt{#f} if successful, and an error message if there is a conflicting tag group definition.
define-void-function ...  
Like define-music-function, but the return value must be the special ‘*unspecified*’ value (i.e., what most Guile functions with “unspecified” value return). Use this when defining functions for executing actions rather than returning values, to keep LilyPond from trying to interpret the return value.

degrees->radians angle-degrees  
Convert the given angle from degrees to radians.

descend-to-context m context [id [mods]]  
Like context-spec-music, but only descending.

determine-split-list evl1 evl2 chord-range  
Event lists evl1 and evl2 should be ascending. chord-range is a pair of numbers (min . max) defining the distance in steps between notes that may be combined into a chord or unison.

determine-string-fret-finger context notes specified-info rest  
Determine string numbers and frets for playing notes as a chord, given specified information specified-info. specified-info is a list with two list elements, specified strings defined-strings and specified fingerings defined-fingers. Only a fingering of 0 will affect the fret selection, as it specifies an open string. If defined-strings is ‘()', the context property defaultStrings is used as a list of defined strings. Looks for predefined fretboards if predefinedFretboardTable is not #f. If rest is present, it contains the FretBoard grob, and a fretboard gets created. Otherwise, a list of (string fret finger) lists is returned.

ly:dimension? d  
Is d a dimension? Used to distinguish length variables from normal numbers.

ly:dir? s  
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.

dir-basename file rest ...  
Strip suffixes in rest, but leave directory component for file.

ly:directed direction magnitude  
Calculate 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  
Stop the dispatcher to listening to events from from.

ly:dispatcher? x  
Is x a smob of class Dispatcher?

display-lily-music expr [port]  
Display the music expression expr using LilyPond syntax.

display-music music [port]  
Display music, not done with music-map for clarity of presentation.

display-scheme-music obj [port]  
Display obj, typically a music expression, in a friendly fashion, which often can be read back in order to generate an equivalent expression.
dodecaphonic-no-repeat-rule context pitch barnum

An accidental rule that typesets an accidental before every note (just as in the dodecaphonic accidental style) except if the note is immediately preceded by a note with the same pitch. This is a common accidental style in contemporary notation.

ly:duration? x
Is x a smob of class Duration?

ly:duration<? p1 p2
Is p1 shorter than p2?

ly:duration->string dur
Convert dur to a string.

ly:duration-compress dur factor
Compress dur by rational factor.

ly:duration-dot-count dur
Extract the dot count from dur.

duration-dot-factor dotcount
Given a count of the dots used to extend a musical duration, return the numeric factor by which they increase the duration.

ly:duration-factor dur
Extract the compression factor from dur. Return it as a pair.

ly:duration-length dur
The length of the duration as a moment.

duration-length dur
Return the overall length of a duration, as a number of whole notes. (Not to be confused with ly:duration-length, which returns a less useful Moment object.)

duration-line::calc grob
Return list of values needed to print a stencil for DurationLine.

duration-line::print grob
Return the stencil of DurationLine.

ly:duration-log dur
Extract the duration log from dur.

duration-log-factor lognum
Given a logarithmic duration number, return the length of the duration, as a number of whole notes.

ly:duration-scale dur
Extract the compression factor from dur. Return it as a rational.

duration-visual dur
Given a duration object, return the visual part of the duration (base note length and dot count), in the form of a duration object with non-visual scale factor 1.

duration-visual-length dur
Given a duration object, return the length of the visual part of the duration (base note length and dot count), as a number of whole notes.
Chapter 4: Scheme functions

**dynamic-text-spanner::before-line-breaking grob**  
Monitor left bound of DynamicTextSpanner for absolute dynamics. If found, ensure DynamicText does not collide with spanner text by changing 'attach-dir and 'padding. Reads the 'right-padding property of DynamicText to fine-tune space between the two text elements.

**ly:effective-prefix**  
Return effective prefix. For example, if LilyPond Scheme files are stored in directory /foo/bar/scm and PS files in /foo/bar/ps, the effective prefix is /foo/bar.

**elbowed-hairpin coords mirrored?**  
Create hairpin based on a list of coords in (cons x y) form. x is the portion of the width consumed for a given line and y is the portion of the height. For example, '((0 . 0) (0.3 . 0.7) (0.8 . 0.9) (1.0 . 1.0)) means that at the point where the hairpin has consumed 30% of its width, it must be at 70% of its height. Once it is to 80% width, it must be at 90% height. It finishes at 100% width and 100% height. If coods does not begin with '(0 . 0) the final hairpin may have an open tip. For example '(0 . 0.5) will cause an open end of 50% of the usual height.

*mirrored?* indicates if the hairpin is mirrored over the y axis or if just the upper part is drawn.

Returns a function that accepts a hairpin grob as an argument and draws the stencil based on its coordinates.

```scheme
#(define simple-hairpin  
  (elbowed-hairpin '((0 . 0)(1.0 . 1.0)) #t))

\relative c' {  
  \override Hairpin #'stencil = #simple-hairpin  
  a\p<a a a\f  }
```

**ellipse-stencil stencil thickness x-padding y-padding**  
Add an ellipse around stencil, padded by the padding pair, producing a new stencil.

**end-broken-spanner? spanner**  
Is spanner broken and the last of its broken siblings? See also unbroken-or-last-broken-spanner?.

**ly:engraver-announce-end-grob engraver grob cause**  
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**  
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:engraver-make-item engraver grob-name cause**  
Same as ly:engraver-make-grob, but always create a grob with the Item class. This is useful when the same grob definition is used to create grobs of differing classes.

**ly:engraver-make-spanner engraver grob-name cause**  
Same as ly:engraver-make-grob, but always create a grob with the Spanner class. This is useful when the same grob definition is used to create grobs of differing classes.
ly: engraver-make-sticky engraver grob-name host cause
Utility function to create a grob sticking to another grob. This acts like either ly: engraver-make-item or ly: engraver-make-spanner, depending on the class of the host. Additionally, the host is made the parent of the newly created sticky grob on the y axis and, for items, on the x axis. Sticky spanners take their bounds from their host and their end is announced with the end of the host.

Sticky grobs must have the sticky-grob-interface interface, see Section “sticky-grob-interface” in Internals Reference.

ly:error str rest
A Scheme callable function to issue the error str. The error is formatted with format; rest holds the formatting arguments (if any).

eval-carefully symbol module default ...
Check whether all symbols in expression symbol are reachable in module module. In that case evaluate, otherwise print a warning and set an optional default.

ly:event? obj
Is obj a proper (non-rhythmic) Event object?

event-chord-notes event-chord
Return a list of all notes from event-chord.

event-chord-pitches event-chord
Return a list of all pitches from event-chord.

event-chord-reduce music
Reduce event chords in music to their first note event, retaining only the chord articulations. Returns the modified music.

event-chord-wrap! music
Wrap isolated rhythmic events and non-postevent events in music inside of an EventChord. Chord repeats ‘q’ are expanded using the default settings of the parser.

ly:event-deep-copy m
Copy m and all sub-expressions of m.

ly:event-property sev sym val
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
Set property sym in event ev to val.

expand-repeat-chords! event-types music
Walk through music and fill repeated chords (notable by having a duration in duration) with the notes from their respective predecessor chord.

expand-repeat-notes! music
Walk through music and give pitchless notes (not having a pitch in pitch or a drum type in drum-type) the pitch(es) from the predecessor note/chord if available.
ly:expect-warning \textit{str rest} \hfill [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 is shown. The message should be translated with \texttt{(\_ \ldots)} and changing parameters given after the format string.

\textbf{extract-beam-exceptions} \textit{music} \hfill [Function]
Create a value useful for setting beamExceptions from \textit{music}.

\textbf{extract-music} \textit{music pred?} \hfill [Function]
Return a flat list of all music matching \textit{pred?} inside of \textit{music}, not recursing into matches themselves.

\textbf{extract-named-music} \textit{music music-name} \hfill [Function]
Return a flat list of all music named \textit{music-name} (either a single event symbol or a list of alternatives) inside of \textit{music}, not recursing into matches themselves.

\textbf{ly:extract-subfont-from-collection} \textit{collection-file-name idx subfont-file-name} \hfill [Function]
Extract the subfont of index \textit{idx} in TrueType collection (TTC) or OpenType/CFF collection (OTC) file \textit{collection-file-name} and write it to file \textit{subfont-file-name}.

\textbf{extract-typed-music} \textit{music type} \hfill [Function]
Return a flat list of all music with \textit{type} (either a single type symbol or a list of alternatives) inside of \textit{music}, not recursing into matches themselves.

\textbf{ly:find-file} \textit{name} \hfill [Function]
Return the absolute file name of \textit{name}, or \texttt{#f} if not found.

\textbf{find-named-props} \textit{prop-name grob-descriptions} \hfill [Function]
Used by \texttt{\magnifyMusic} and \texttt{\magnifyStaff}. If \textit{grob-descriptions} is equal to the all-grob-descriptions alist (defined in \texttt{scm/define-grobs.scm}), this finds all grobs that can have a value for the \textit{prop-name} property, and return them as a list in the following format:

\begin{verbatim}
  '((grob prop-name)
   (grob prop-name)
   ...)
\end{verbatim}

\textbf{find-pitch-entry} \textit{keysig pitch accept-global accept-local} \hfill [Function]
Return the first entry in \textit{keysig} that matches \textit{pitch} by notename and octave. Alteration is not considered. \texttt{accept-global} states whether key signature entries should be included. \texttt{accept-local} states whether local accidentals should be included. If no matching entry is found, \texttt{#f} is returned.

\textbf{finger-glide::print} \textit{grob} \hfill [Function]
The stencil printing procedure for grob \texttt{FingerGlideSpanner}. Depending on the grob property \texttt{style} several forms of appearance are printed. Possible settings for grob property \texttt{style} are zigzag, trill, dashed-line, dotted-line, stub-left, stub-right, stub-both, bow, none and line, which is the default.

\textbf{first-assoc} \textit{keys lst} \hfill [Function]
Return first successful assoc of key from \textit{keys} in \textit{lst}.

\textbf{first-broken-spanner?} \textit{spanner} \hfill [Function]
Is \textit{spanner} broken \textit{and} the first of its broken siblings? See also \texttt{unbroken-or-first-broken-spanner?}. 
first-member *members lst*  
Return first successful member (of member) from members in lst.

flat-flag *grob*  
Flat flag style. The angles of the flags are both 0 degrees.

flatten-list *x*  
Unnest list.

flip-stencil *axis stil*  
Flip stencil stil in the direction of axis. Value X (or 0) for axis flips it horizontally. Value Y (or 1) flips it vertically. stil is flipped in place; its position, the coordinates of its bounding box, remains the same.

fold-some-music *pred? proc init music*  
This works recursively on music like fold does on a list, calling '('pred? music') on every music element. If #f is returned for an element, it is processed recursively with the same initial value of 'previous', otherwise '('proc music previous') replaces 'previous' and no recursion happens. The top music is processed using init for 'previous'.

ly:font-config-add-directory *dir*  
Add directory dir to FontConfig.

ly:font-config-add-font *font*  
Add font font to FontConfig.

ly:font-config-display-fonts  
Dump a list of all fonts visible to FontConfig.

ly:font-config-get-font-file *name*  
Get the file for font name, as found by FontConfig.

ly:font-design-size *font*  
Given the font metric font, return the design size, relative to the current output-scale.

ly:font-file-name *font*  
Given the font metric font, return the corresponding file name.

ly:font-get-glyph *font name*  
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*  
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*  
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 smob of class Font_metric?

ly:font-name  font
[Function]
Given the font metric font, return the corresponding name.

font-name-split  font-name
[Function]
Return (font-name . design-size) from font-name string or #f.

ly:font-sub-fonts  font
[Function]
Given the font metric font of an OpenType font, return the names of the subfonts within
font.

for-some-music  stop? music
[Function]
Walk through music, process all elements calling stop? and only recurse if this returns #f.

ly:format  str rest
[Function]
LilyPond specific format function, 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.

format-segno-mark-considering-bar-lines  segno-number context
[Function]
When bar lines incorporate segni, print no mark for the first segno because that would be
redundant. Print the usual marks for later segni to avoid ambiguity.

fret->pitch  fret
[Function]
Calculate a pitch given fret for the harmonic.

fret-parse-terse-definition-string  props definition-string
[Function]
Parse a fret diagram string that uses terse syntax; return a pair containing props, modified
to include the string-count determined by definition-string, and a fret indication list with the
appropriate values.

function-chain  arg function-list
[Function]
Apply a list of functions in function-list to arg. Each element of function-list is structured
(cons function '(arg2 arg3 ...)). If function takes arguments besides arg, they are pro-
vided in function-list. Example:

(function-chain 1 `((,+ 1) (,- 2) (+ 3) (/)))
⇒ 1/3

generate-crop-stencil  paper-book
[Function]
Returns a stencil for the cropped output of the given Paper_book

generate-preview-stencil  paper-book
[Function]
Returns a stencil for a preview of given Paper_book
ly:generic-bound-extent  \textit{grob common}  
Determine the extent of \textit{grob} relative to \textit{common} along the x axis, finding its extent as a bound when it a has \texttt{bound-alignment-interfaces} property list set and otherwise the full extent.

ly:get-all-function-documentation  
Get a hash table with all LilyPond Scheme extension functions.

ly:get-all-translators  
Return a list of all translator objects that may be instantiated.

get-bound-note-heads \textit{spanner}  
Take a spanner grob and return a pair containing all note heads of the initial starting and the final \texttt{NoteColumn}.

ly:get-cff-offset \texttt{font-file-name idx}  
Get the offset of the ‘CFF’ table for \texttt{font-file-name}, returning it as an integer. The optional \texttt{idx} argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of \texttt{idx} is 0.

get-chord-shape \texttt{shape-code tuning base-chord-shapes}  
Return the chord shape associated with \texttt{shape-code} and \texttt{tuning} in the hash-table \texttt{base-chord-shapes}.

ly:get-context-mods \texttt{contextmod}  
Returns the list of context modifications stored in \texttt{contextmod}.

ly:get-font-format \texttt{font-file-name idx}  
Get the font format for \texttt{font-file-name}, returning it as a symbol. The optional \texttt{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 \texttt{idx} is 0.

ly:get-option \texttt{var}  
Get a global option setting.

get-postscript-bbox \texttt{string}  
Extract the bounding box from \texttt{string}, or return \#f if not present.

ly:get-spacing-spec \texttt{from-scm to-scm}  
Return the spacing spec going between the two given grobs, \texttt{from-scm} and \texttt{to-scm}.

get-tweakable-music \texttt{mus}  
When tweaking music, return a list of music expressions where the tweaks should be applied. Relevant for music wrappers and event chords.

glyph-flag \texttt{flag-style}  
Simulates the default way of generating flags: Look up glyphs \texttt{flags.style[ud][1234]} from the feta font and use it for the flag stencil.

ly:grob? \texttt{x}  
Is \texttt{x} a smob of class \texttt{Grob}?

grob::all-objects \texttt{grob}  
Return a list of the names and contents of all properties having type \texttt{ly:grob?} or \texttt{ly:grob-array?} for all interfaces supported by \texttt{grob grob}.
grob::compose-function  
Create a callback entity `func` to be stored in a grob property, based on the grob property data `data` (which can be plain data, a callback itself, or an unpure-pure container).
Function or unpure-pure container `func` accepts a grob and a value and returns another value. Depending on the type of `data`, `func` is used for building a grob callback or an unpure-pure container.

```
grob::display-objects grob
```
Display all objects stored in properties of grob `grob`.

```
grob::inherit-parent-property axis property default ...
grob callback generator for inheriting a `property` from an `axis` parent, defaulting to `default` if there is no parent or the parent has no setting.
```

```
grob::name grob
```
Return the name of the grob `grob` as a symbol.

```
grob::offset-function func data rest ...
grob callback generator for building a grob callback or an unpure-pure container.
```
Function `func` accepts a grob and `data` and returns another value.
Depending on the type of `data`, `func` is used for building a grob callback or an unpure-pure container.

If `data` is `#f` or `'(())`, it is not included in the sum.

```
grob::relay-other-property property
```
 grob callback generator for returning the value of another property, which is identified by the symbol `property`.

```
grob::rhythmic-location grob
```
Return a pair consisting of the measure number and moment within the measure of grob `grob`.

```
grob::unpure-Y-extent-from-stencil pure-function
```
The unpure height will come from a stencil whereas the pure height will come from `pure-function`.

```
grob::when grob
```
Return the global timestep (a `Moment`) of grob `grob`.

```
ly:grob-alist-chain grob global
```
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
```
Is `x` a smob of class `Grob_array`?

```
ly:grob-array->list grob-arr
```
Return the elements of `grob-arr` as a Scheme list.

```
ly:grob-array-length grob-arr
```
Return the length of `grob-arr`.

```
ly:grob-array-ref grob-arr index
```
Retrieve the `index`th element of `grob-arr`.
ly:grob-basic-properties grob
   Get the immutable properties of grob.

ly:grob-chain-callback grob proc sym
   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
   Find the common refpoint of grob and other for axis.

ly:grob-common-refpoint-of-array grob others axis
   Find the common refpoint of grob and others (a grob-array) for axis.

ly:grob-default-font grob
   Return the default font for grob grob.

ly:grob-extent grob refp axis
   Get the extent in axis direction of grob relative to the grob refp.

ly:grob-get-vertical-axis-group-index grob
   Get the index of the vertical axis group the grob grob belongs to; return -1 if none is found.

ly:grob-interfaces grob
   Return the interfaces list of grob grob.

ly:grob-layout grob
   Get \layout definition from grob grob.

ly:grob-list->grob-array grob-list
   Convert a Scheme list of grobs to a grob array.

ly:grob-object grob sym val
   Return the value of a pointer in grob grob of property sym. When sym is undefined in grob, it returns val if specified or '() (end-of-list) otherwise. The kind of properties this taps into differs from regular properties. It is used to store links between grobs, either grobs or grob arrays. For instance, a note head has a stem property, the stem grob it belongs to. Just after line breaking, all those grobs are scanned and replaced by their relevant broken versions when applicable.

ly:grob-original grob
   Return the unbroken original grob of grob, grob may be an item or spanner.

ly:grob-parent grob axis def
   Get the parent of grob. axis is 0 for the x axis, 1 for the y axis. If grob has no parent on this axis (yet), return def, or '()' if def is not specified.

ly:grob-pq<? a b
   Compare two grob priority queue entries. This is an internal function.

ly:grob-properties? x
   Is x a smob of class Grob_properties?

ly:grob-property grob sym val
   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
   Return the value for property sym of grob, but do not process callbacks.
ly:grob-pure-height  grob refp beg end val
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
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
Get the coordinate in axis direction of grob relative to the grob refp.

ly:grob-robust-relative-extent  grob refp axis
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
Compare two grobs by script priority. For internal use.

ly:grob-set-nested-property!  grob symlist val
Set nested property symlist in grob grob to value val.

ly:grob-set-object!  grob sym val
Set sym in grob grob to value val.

ly:grob-set-parent!  grob axis parent-grob
Set parent-grob as the parent of grob grob in axis axis.

ly:grob-set-property!  grob sym val
Set sym in grob grob to value val.

ly:grob-spanned-column-rank-interval  grob
Return 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
Return the y position of sg relative to the staff.

ly:grob-suicide!  grob
Kill grob.

ly:grob-system  grob
Return the system grob of grob.

grob-transformer  property func
Create an override value good for applying func to either pure or unpure values. func is called with the respective grob as first argument and the default value (after resolving all callbacks) as the second.

ly:grob-translate-axis!  grob d a
Translate grob on axis a over distance d.

ly:grob-vertical<?  a b
Does a lie above b on the page?

ly:gulp-file  name size
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:gulp-file-utf8 name size  
Read size characters from the file name, and return its contents in a string decoded from UTF-8. 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  
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  
Return a list of keys in tab.

headers-property-alist-chain headers  
Take a list of \header blocks (Guile modules). Return an alist chain containing all of their bindings where the names have been prefixed with header:. This alist chain is suitable for interpreting a markup in the context of these headers.

hook-stencil x y staff-space thick blot grob  
Return a hook stencil where x determines the horizontal position and y determines the basic vertical position. The final stencil is adjusted vertically using staff-space, which is StaffSymbol’s staff space, and uses blot, which is the current ‘blot-diameter. The stencil’s thickness is usually taken from grob ‘details, thick serves as a fallback value.

ly:in-event-class? ev cl  
Does event ev belong to event class cl?

ly:inch num  
num inches.

ly:input-both-locations sip  
Return input location in sip as
(file-name first-line first-column last-line last-column)

ly:input-file-line-char-column sip  
Return input location in sip as (file-name line char column).

ly:input-location? x  
Is x a smob of class Input?

ly:input-message sip msg rest  
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  
Print msg as a GNU compliant warning message, pointing to the location in sip. msg is interpreted similar to format’s argument, using rest.

interpret-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:interpret-music-expression  \texttt{mus ctx} \quad \textbf{[Function]}
Interpret the music expression \texttt{mus} in the global context \texttt{ctx}. The context is returned in its final state.

interval-center x \quad \textbf{[Function]}
Center the number pair \texttt{x}, if an interval.

interval-index \texttt{interval dir} \quad \textbf{[Function]}
Interpolate \texttt{interval} between between left (\texttt{dir=-1}) and right (\texttt{dir=+1}).

interval-length x \quad \textbf{[Function]}
Length of the number pair \texttt{x}, if an interval.

ly:intlog2 d \quad \textbf{[Function]}
The 2-logarithm of $1/d$.

invalidate-alterations \texttt{context} \quad \textbf{[Function]}
Invalidate alterations in \texttt{context}.

Elements of 'localAlterations corresponding to local alterations of the key signature have the form '(((octave . notename) . (alter barnum . end-mom)). Replace them with a version where \texttt{alter} is set to 'clef to force a repetition of accidentals.

Entries that conform with the current key signature are not invalidated.

ly:item? g \quad \textbf{[Function]}
Is \texttt{g} an Item object?

item::extra-spacing-height-including-staff \texttt{grob} \quad \textbf{[Function]}
Return a value for extra-spacing-height that augments the extent of the grob to the extent of the staff.

ly:item-break-dir it \quad \textbf{[Function]}
The break status direction of item \texttt{it}. \texttt{-1} means end of line, \texttt{0} unbroken, and \texttt{1} beginning of line.

ly:item-get-column it \quad \textbf{[Function]}
Return the PaperColumn or NonMusicalPaperColumn associated with this Item.

ly:iterator? x \quad \textbf{[Function]}
Is \texttt{x} a smob of class Music_iterator?

layout-line-thickness \texttt{grob} \quad \textbf{[Function]}
Get the line thickness of the \texttt{grob}'s corresponding layout.

layout-set-absolute-staff-size \texttt{sz} \quad \textbf{[Function]}
Set the absolute staff size inside of a \texttt{\layout{}} block. \texttt{sz} is in points.

layout-set-staff-size \texttt{sz} \quad \textbf{[Function]}
Set the staff size inside of a \texttt{\layout{}} block. \texttt{sz} is in points.

ly:length \texttt{x y} \quad \textbf{[Function]}
Calculate magnitude of given vector. With one argument, \texttt{x} is a number pair indicating the vector. With two arguments, \texttt{x} and \texttt{y} specify the respective coordinates.

ly:lily-lexer? \texttt{x} \quad \textbf{[Function]}
Is \texttt{x} a smob of class LilyLexer?
ly:lily-parser? x
Is x a smob of class Lily_parser?

lilypond-main files
Entry point for LilyPond.

lilypond-version-outdated? file-version lily-version
Is file-version outdated compared to lily-version? This is defined as a version that is from a lower release series (corresponding to the first two numbers of the version) or a version from the same unstable release series (odd minor version number) with a lower patch level (third number). A stable version from the same series does not count as outdated because compatibility is preserved.

ly:line-interface::line grob startx starty endx endy
Make a line using layout information from grob grob.

list-insert-separator lst between
Create new list, inserting between between elements of lst.

list-join lst intermediate
Put intermediate between all elements of lst.

ly:listened-event-class? disp cl
Does disp listen to any event type in the list cl?

ly:listened-event-types disp
Return a list of all event types that disp listens to.

ly:listener? x
Is x a smob of class Listener?

lookup-markup-command code
Return (function . signature) for a markup command code, or return #f.

lyric-hyphen::vaticana-style grob
Draw a LyricHyphen grob as needed for Gregorian chant in Editio Vaticana style, that is, apply it once, flush-left. If the text property of LyricHyphen is set, print this markup. If the property is not set, use a hyphen character.

lyric-text::print grob
Allow interpretation of tildes as lyric tieing marks.

make-accidental-dodecaphonic-rule octaveness laziness
Variation on function make-accidental-rule that creates a dodecaphonic accidental rule.

make-accidental-rule octaveness laziness
Create an accidental rule that makes its decision based on the octave of the note and a laziness value.

octaveness is either 'same-octave or 'any-octave and defines whether the rule should respond to accidental changes in other octaves than the current. 'same-octave is the normal way to typeset accidentals – an accidental is made if the alteration is different from the last active pitch in the same octave. 'any-octave looks at the last active pitch in any octave.
laziness states over how many bars an accidental should be remembered. 0 is the default – accidental lasts over 0 bar lines, that is, to the end of current measure. A positive integer means that the accidental lasts over that many bar lines. -1 is ‘forget immediately’, that is, only look at key signature. #t is ‘forever’. 
ly:make-book  
Make a \textit{book} of \textit{paper} and \textit{header} (which may be \texttt{#f} as well) containing \texttt{\scores}.

ly:make-book-part  
Make a \textit{bookpart} containing \texttt{\scores}.

make-bow-stencil  
Create a bow stencil. It starts at point \texttt{start}, ends at point \texttt{stop}. \textit{thickness} is the thickness of the bow. The higher the value of number \textit{angularity}, the more angular the shape of the bow. \textit{bow-height} determines the height of the bow. \textit{orientation} determines whether the bow is concave or convex. Both variables are supplied to support independent usage.

Done by calculating a horizontal unit bow first, then moving all control points to the correct positions. Limitation: s-curves are currently not supported.

make-c-time-signature-markup  
Make markup for the ‘C’ time signature style.

make-circle-stencil  
Make a circle of radius \texttt{radius} and thickness \texttt{thickness}.

make-clef-set  
Generate the clef setting commands for a clef with name \texttt{clef-name}.

make-connected-line  
Take a list of points, \texttt{points}. Return a line connecting \texttt{points}, using \texttt{ly:line-interface::line} and getting layout information from \texttt{grob}.

make-connected-path-stencil  
Make a connected path described by the list \texttt{pointlist}, beginning at point (0, 0), with thickness \texttt{thickness}, and scaled by \texttt{x-scale} in the x direction and \texttt{y-scale} in the y direction. \texttt{connect} and \texttt{fill} are boolean arguments that specify whether the path should be connected or filled, respectively.

ly:make-context-mod  
Create a context modification, optionally initialized via the list of modifications \texttt{mod-list}.

make-cue-clef-set  
Generate the clef setting commands for a cue clef with name \texttt{clef-name}.

make-cue-clef-unset  
Reset the clef settings for a cue clef.

ly:make-dispatcher  
Return a newly created dispatcher.

ly:make-duration  
Make a duration. \texttt{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 \texttt{dotcount}.

The duration factor is optionally given by integers \texttt{num} and \texttt{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.
make-duration-of-length moment
   Make duration of the given moment length.

make-ellipse-stencil x-radius y-radius thickness fill
   Make an ellipse of x radius x-radius, y radius y-radius, and thickness thickness with fill defined by fill.

make-engraver ...
   Like make-translator, but create an engraver, i.e., the resulting translator is only run in layout output and ignored in MIDI.

make-filled-box-stencil xext yext
   Make a filled box.

ly:make-global-context output-def
   Set up a global interpretation context, using the output block output-def. The context is returned.

ly:make-global-translator global
   Create a translator group and connect it to the global context global. The translator group is returned.

make-glyph-time-signature-markup style fraction
   Make markup for a symbolic time signature of the form timesig.<style><numerator><denominator>, for example 'timesig.mensural34'. If the music font does not have a glyph for the requested style and fraction, issue a warning and make a numbered time signature instead.

ly:make-grob-properties alist
   Package the given property list alist in a grob property container stored in a context property with the name of a grob.

make-grob-property-override grob gprop val
   Make a Music expression that overrides gprop to val in grob. This is a temporary \override, making it possible to \revert to any previous value afterwards.

make-grob-property-revert grob gprop
   Revert the grob property gprop for grob.

make-grob-property-set grob gprop val
   Make a Music expression that overrides a gprop to val in grob. Does a pop first, i.e., this is not a temporary \override.

make-harmonic mus
   Convert music variable mus to harmonics.

make-line-stencil width startx starty endx endy
   Make a line stencil of given line width and set its extents accordingly.

ly:make-listener callback
   This is a compatibility wrapper for creating a 'listener' for use with ly:add-listener from a callback taking a single argument. Since listeners are equivalent to callbacks, this is no longer needed.

make-modal-inverter around to scale
   Wrapper function for inverter-factory.
make-modal-transposer from to scale  
  Wrapper function for transposer-factory.

ly:make-moment m g gn gd  
  Create a 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  
  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.

make-music name music-properties ...  
  Create a music object of given name, and set its properties according to music-properties, a list of alternating property symbols and values. Example:

    (make-music 'OverrideProperty
      'symbol 'Stem
      'grob-property 'thickness
      'grob-value (* 2 1.5))

  Instead of a successive symbol and value, an entry in the list may also be an alist or a music object in which case its elements, respectively its mutable property list (properties not inherent to the type of the music object), are taken.

  The argument list will be interpreted left to right, so later entries override earlier ones.

ly:make-music-function signature func  
  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:rating? predicates or other type predicates. Its car is the syntax function to call.

ly:make-music-relative! music pitch  
  Make music relative to pitch, return final pitch.

ly:make-output-def  
  Make an output definition.

make-oval-stencil x-radius y-radius thickness fill  
  Make an oval from two Bézier curves, of x radius x-radius, y radius y-radius, and thickness thickness with fill defined by fill.

ly:make-page-label-marker label  
  Return page marker with label label.

ly:make-page-permission-marker symbol permission  
  Return page marker with page breaking and turning permissions.

ly:make-pango-description-string chain size  
  Make a PangoFontDescription string for the property alist chain at size size.
ly:make-paper-outputter\textit{ port\ list\ default-callback} \hfill [Function]

Create an outputter dumping to\textit{ port}.\textit{ list\ should\ map symbols to procedures. See file output-ps.scm for an example. If\ default-callback\ is\ given, it is called for unsupported expressions.}

make-part-combine-context-changes\textit{ state-machine\ split-list} \hfill [Function]

Generate a sequence of part combiner context changes from a split list.

make-part-combine-marks\textit{ state-machine\ split-list} \hfill [Function]

Generate a sequence of part combiner events from a split list.

make-partial-ellipse-stencil\textit{ x-radius\ y-radius\ start-angle\ end-angle} \hfill [Function]

\textit{thick\ connect\ fill}

Create an elliptical arc.\textit{ x-radius\ is the x radius of the arc.\textit{ y-radius\ is the y radius of the arc.\textit{ start-angle\ is the starting angle of the arc (in degrees).\textit{ end-angle\ is the ending angle of the arc (in degrees).\textit{ thick\ is the thickness of the line.\textit{ connect\ is a boolean flag indicating whether the end should be connected to the start by a line.\textit{ fill\ is a boolean flag indicating whether the shape should be filled.}

make-path-stencil\textit{ path\ thickness\ x-scale\ y-scale\ fill\ #:line-cap-style\ #:line-join-style\ line-join-style} \hfill [Function]

Make a stencil based on the path described by the list\textit{ path}, with thickness\textit{ thickness}, and scaled by\textit{ x-scale} in the x direction and\textit{ y-scale} in the y direction (the difference with scaling the resulting stencil using\textit{ ly:stencil-scale} is that this scaling does not change the thickness).\textit{ fill} is a boolean argument that specifies whether the path should be filled. Valid path commands are

\begin{itemize}
  \item moveto
  \item rmoveto
  \item lineto
  \item rlineto
  \item curveto
  \item rcurveto
  \item closepath
\end{itemize}

and their standard SVG single-letter equivalents

\begin{itemize}
  \item M
  \item m
  \item L
  \item l
  \item C
  \item c
  \item Z
  \item z
\end{itemize}

make-performer \ldots \hfill [Macro]

Like\textit{ make-translator}, but create a performer, i.e., the resulting translator is only run in MIDI and ignored in layout output. Scheme performers do not support acknowledgers and process-acknowledged.

ly:make-pitch\textit{ octave\ note\ alter} \hfill [Function]

Make a pitch.\textit{ octave} is specified by an integer, zero for the octave containing middle C.\textit{ note} is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional\textit{ alter} is a rational number of 200-cent whole tones for alteration.

ly:make-prob\textit{ type\ init\ rest} \hfill [Function]

Create a\textit{ Prob} object.

make-relative \ldots \hfill [Macro]

The list of pitch or music variables in\textit{ variables} is used as a sequence for creating relativable music from\textit{ music}.

When the constructed music is used outside of\textit{ \relative}, it just reflects plugging in the\textit{ variables} into\textit{ music}.

The action inside of\textit{ \relative}, however, is determined by first relativizing the surrogate\textit{ reference} with the variables plugged in and then using the variables relativized as a side effect of relativizing\textit{ reference} for evaluating\textit{ music}.  


Since pitches don’t have the object identity required for tracing the effect of the reference call, they are replaced only for the purpose of evaluating reference with simple pitched note events.

The surrogate reference expression has to be written with that in mind. In addition, it must not contain copies of music that is supposed to be relativized but rather the originals. This includes the pitch expressions. As a rule, inside of #\{\ldots\#\} variables must only be introduced using #, never via the copying construct $. The reference expression will usually just be a sequential or chord expression naming all variables in sequence, implying that following music will be relativized according to the resulting pitch of the last or first variable, respectively.

Since the usual purpose is to create more complex music from general arguments and since music expression parts must not occur more than once, one does generally need to use copying operators in the replacement expression music when using an argument more than once there. Using an argument more than once in reference, in contrast, does not make sense.

There is another fine point to mind: music must only contain freshly constructed elements or copied constructs. This will be the case anyway for regular LilyPond code inside of #\{\ldots\#\}, but any other elements (apart from the variables themselves which are already copied) must be created or copied as well.

The reason is that it is usually permitted to change music in-place as long as one does a ly:music-deep-copy on it, and such a copy of the whole resulting expression will not be able to copy variables/values inside of closures where the information for relativization is being stored.

```
make-repeat name times main alts
Create a repeat music expression, with all properties initialized properly.
```

```
ly:make-rotation angle center
Make a transform rotating by angle in degrees. If center is given as a pair of coordinates, it is the center of the rotation, otherwise the rotation is around (0, 0).
```

```
ly:make-scale steps
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-scaling scale scaley
Create a scaling transform from argument scale and optionally scaley. When both arguments are given, they must be real and give the scale in x and y direction. If only scale is given, it may also be complex to indicate a scaled rotation in the manner of complex number rotations, or a pair of reals for specifying different scales in x and y direction like with the first calling convention.
```

```
ly:make-score music
Return score with music encapsulated in it.
```

```
make-semitone->pitch pitches
Convert pitches, an unordered list of note values covering (after disregarding octaves) all absolute pitches in need of conversion, into a function converting semitone numbers (absolute pitch missing enharmonic information) back into note values.
```

For a key signature without accidentals

```
c cis d es e fis g gis a bes b
```

might be a good choice, covering Bb major to A major and their parallel keys, and melodic/harmonic C minor to A minor.
Chapter 4: Scheme functions

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.

make-stencil-boxer  *thickness padding callback*  [Function]
Return function that adds a box around the grob passed as argument.

make-stencil-circler  *thickness padding callback*  [Function]
Return function that adds a circle around the grob passed as argument.

ly:make-stream-event  *cl proplist*  [Function]
Create a stream event of class *cl* with the given mutable property list.

make-tmpfile  *dir*  [Function]
Return a temporary file (as a Scheme port). If *dir* is #f, a file in the directory given by the
environment variable *$TMPDIR* is created.

ly:make-transform  *xx yx xy yy x0 y0*  [Function]
Create a transform. Without options, it is the identity transform. Given four arguments
*xx*, *yx*, *xy*, and *yy*, it is a linear transform. Given six arguments (with *x0* and *y0* last), it is an
affine transform.

Transforms can be called as functions on other transforms (concatenating them) or on
points given either as complex number or real number pair. See also ly:make-rotation,

ly:make-translation  *x y*  [Function]
Make a transform translating by *x* and *y*. If only *x* is given, it can also be a complex number
or a pair of numbers indicating the offset to use.

make-translator ...  [Macro]
Helper macro for creating Scheme translators usable in both ‘\midi’ and ‘\layout’.
The usual form for a translator is an association list (or alist) mapping symbols to either
anonymous functions or to another such alist.

make-translator accepts forms where the first element is either an argument list starting
with the respective symbol, followed by the function body (comparable to the way define
is used for defining functions), or a single symbol followed by subordinate forms in the same
manner. You can also just make an alist pair literally (the ‘car’ is quoted automatically) as
long as the unevaluated ‘cdr’ is not a pair. This is useful if you already have defined your
engraver functions separately.

Symbols mapping to a function would be initialize, start-translation-timestep,
pre-process-music, process-music, stop-translation-timestep, and finalize. Symbols
mapping to another alist specified in the same manner are listeners with the subordinate
symbols being event classes.

A template for writing a translator with all methods is:

(lambda (context)
  (let (local-variables ...)
    (make-translator ...))
((initialize translator) ...) 
((start-translation-timestep translator) ...) 
(listeners 
  ((event-class-1 translator event) ...) 
  ((event-class-2 translator event #:once) ...) 
  (process-music translator) ...
)(acknowledgers 
  ((grob-interface-1 translator grob source-translator) ...) 
  ((grob-interface-2 translator grob source-translator) ...) 
  (process-acknowledged translator) ...
  (stop-translation-timestep translator) ...) 
  ((finalize translator) ...)))

This can be used as the argument to \consists.

For listeners, a special feature is available: the argument list of a listener can be terminated with the keyword #:once. This makes for a listener that is only ever triggered once per time step. If it receives several events in the same time step, it emits a warning, except if they are all equal (where equality is checked recursively, with equal?).

(make-transparent-box-stencil xext yext) [Function]
Make a transparent box.

(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.

(map-selected-alist-keys function keys alist) [Function]
Return alist with function applied to all of the values in list keys. Example:
(map-selected-alist-keys - '(a b) '([(a . 1) (b . -2) (c . 3) (d . 4)])
⇒ ((a . -1) (b . 2) (c . 3) (d . 4))

(map-some-music map? music) [Function]
Walk through music, transform all elements calling map? and only recurse if this returns #f. elements or articulations that are not music expressions are discarded: this allows some amount of filtering.

(map-some-music) may overwrite the original music.

(marked-up-headfoot what-odd what-even) [Function]
Read variables what-odd and what-even from the page’s layout. Interpret either of them as markup, with properties reflecting the variables in the page’s layout and header modules.

(marked-up-title what) [Function]
Read variables what-odd and what-even from the page’s layout. Interpret either of them as markup, with properties reflecting the variables in the page’s layout and header modules.
The markup macro provides a LilyPond-like syntax for building markups using Scheme keywords, replacing \command with #:command. For example, this:

```scheme
\markup { foo
    \raise #0.2 \hbracket \bold bar
    \override #'(baseline-skip . 4)
    \bracket \column { baz bazr bla }
}
```

translates to this:

```scheme
(markup "foo"
    #:raise 0.2 #:hbracket #:bold "bar"
    #:override '(baseline-skip . 4)
    #:bracket #:column ("baz" "bazr" "bla"))
```

\markup->string \m#:layout layout #:props props

Convert a markup or markup list to an approximate string representation. This is useful for, e.g., PDF metadata and MIDI markers.

The optional named layout and props argument are an output definition and a property alist chain, like the ones that are used when interpreting markups.

\markup-command-list? x

Check whether x is a markup command list, i.e., a list composed of a markup list function and its arguments.

\markup-default-to-string-method layout props args ...

The default markup->string handler for markups, used when markup->string encounters a markup that has no special as-string expression defined. This applies markup->string on all markup arguments and joins the results, separating them with spaces.

\markup-lambda ...

Defines and returns an anonymous markup command. Other than not registering the markup command, this is identical to define-markup-command.

\markup-list? arg

Return a true value if x is a list of markups or markup command lists.

\markup-list-lambda ...

Same as markup-lambda but defines a markup list command that, when interpreted, returns a list of stencils instead of a single one.

\measure-counter::text grob

A number for a measure count. Broken measures are numbered in parentheses. When the counter spans several measures (like with compressed multi-measure rests), it displays a measure range.

\mensural-flag grob

Mensural flags: Create the flag stencil by loading the glyph from the font. Flags are always aligned with staff lines, so we need to check the end point of the stem: For stems ending on staff lines, use different flags than for notes between staff lines. The idea is that flags are always vertically aligned with the staff lines, regardless of whether the note head is on a staff line or between two staff lines. In other words, the inner end of a flag always touches a staff line.
ly:message  str  rest
A Scheme callable function to issue the message str. The message is formatted with format; rest holds the formatting arguments (if any).

middle-broken-spanner?  spanner
Is spanner broken and among the middle broken pieces (i.e., neither the first nor the last)?

midi-program  instrument
Return the program of the instrument.

ly:minimal-breaking  paper-book
Break (pages and lines) the Paper_book object paper-book without looking for optimal spacing: stack as many lines on a page before moving to the next one.

ly:mm  num
num mm.

mmrest-of-length  mus
Create a multi-measure rest of exactly the same length as mus.

modern-straight-flag  grob
Modern straight flag style (for composers like Stockhausen, Boulez, etc.). The angles are 18 and 22 degrees and thus smaller than for the ancient style of Bach, etc.

ly:module->alist  mod
Dump the contents of module mod as an alist.

ly:module-copy  dest src
Copy all bindings from module src into dest.

ly:modules-lookup  modules  sym  def
Look up sym in the list modules, returning the first occurrence. If not found, return def or #f if def isn’t specified.

ly:moment?  x
Is x a smob of class Moment?

ly:moment<?  a  b
Compare two moments.

ly:moment-add  a  b
Add two moments.

ly:moment-div  a  b
Divide two moments.

ly:moment-grace  mom
Extract grace timing as a rational number from mom.

ly:moment-grace-denominator  mom
Extract denominator from grace timing.

ly:moment-grace-numerator  mom
Extract numerator from grace timing.

ly:moment-main  mom
Extract main timing as a rational number from mom.
ly:moment-main-denominator mom
    Extract denominator from main timing.

ly:moment-main-numerator mom
    Extract numerator from main timing.

ly:moment-mod a b
    Modulo of two moments.

ly:moment-mul a b
    Multiply two moments.

ly:moment-sub a b
    Subtract two moments.

ly:music? obj
    Is obj a Music object?

music->make-music obj
    Generate an expression that, once evaluated, may return an object equivalent to
    obj, that is, for a music expression, a (make-music ...) form.

music-clone music music-properties ...
    Clone music and set properties according to music-properties, a list of alternating property
    symbols and values:
        (music-clone start-span 'span-direction STOP)
    Only properties that are not overridden by music-properties are actually fully cloned.

ly:music-compress mus scale
    Compress mus by scale.

ly:music-deep-copy m origin
    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
    Compress mus by factor fact, which is a Moment.

ly:music-duration-length mus
    Extract the duration field from mus and return the length.

music-filter pred? music
    Filter out music expressions that do not satisfy pred?.

ly:music-function? x
    Is x a smob of class Music_function?

ly:music-function-extract x
    Return the Scheme function inside x.

ly:music-function-signature x
    Return the function signature inside x.

music-is-of-type? mus type
    Does mus belong to the music class type?
ly:.music-length  
Get the length of music expression mus and return it as a Moment object.

ly:music-list?  
Is lst a list of music objects?

music-map function music
Apply function to music and all of the music it contains.
First it recurses over the children, then the function is applied to music.

ly:music-mutable-properties mus
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
Is x a smob of class Music_output?

music-pitches music
Return a list of all pitches from music.

ly:music-property mus sym val
Return the value for property sym of music expression mus. If no value is found, return val or ’() if val is not specified.

music-selective-filter descend? pred? music
Recursively filter out music expressions that do not satisfy pred?, but refrain from filtering the subexpressions of music that does not satisfy descend?.

music-selective-map descend? function music
Apply function recursively to music, but refrain from mapping subexpressions of music that does not satisfy descend?.

music-separator? m
Is m a separator?

ly:music-set-property! mus sym val
Set property sym in music expression mus to val.

ly:music-start mus
Get the start of music expression mus and return it as a Moment object.

ly:music-transpose m p
Transpose m such that central C is mapped to p. Return m.

music-type-predicate types
Return a predicate function that can be used for checking music to have one of the types listed in types.

neo-modern-accidental-rule context pitch barnum
An accidental rule that typesets an accidental if it differs from the key signature and does not directly follow a note on the same staff line. This rule should not be used alone because it does neither look at bar lines nor different accidentals at the same note name.

no-flag grob
No flag: Simply return empty stencil.
ly:non-fatal-error str rest [Function]
A Scheme callable function to issue the error str. The error is formatted with format; rest holds the formatting arguments (if any). When using this function, some way of signalling the error should be employed in order for the compilation to eventually result in a nonzero return code.

normal-flag grob [Function]
Create a default flag.

not-first-broken-spanner? spanner [Function]
Is spanner broken and not the first of its broken siblings? The name is read “(not first) and broken”.

not-last-broken-spanner? spanner [Function]
Is spanner broken and not the last of its broken siblings? The name is read “(not last) and broken”.

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-extra-source-file filename parser [Function]
Register a file, e.g., an image file, as being needed to compile the current file. This is used for the -dembed-source-code option. A parser may optionally be specified.

ly:note-head::stem-attachment font-metric glyph-name direction [Function]
Get attachment in font-metric for attaching a stem to notehead glyph-name in the direction direction (default UP).

note-name->markup pitch lowercase? [Function]
Return pitch markup for pitch, including accidentals printed as glyphs. If lowercase? is set to false, the note names are capitalized.

note-name->string pitch language ... [Function]
Return pitch string for pitch, without accidentals or octaves. Current input language is used for pitch names, except if an other language is specified.

note-to-cluster music [Function]
Replace NoteEvents by ClusterNoteEvents.

ly:number->string s [Function]
Convert s to a string without generating many decimals.

number-format number-type num custom-format ... [Function]
Print num according to the requested number-type. Choices include arabic, custom, roman-ij-lower, roman-ij-upper, roman-lower (the default), and roman-upper.
For custom, custom-format must be present; it gets applied to num.

offset-fret fret-offset diagram-definition [Function]
Add fret-offset to each fret indication in diagram-definition and return the resulting verbose fret-diagram-definition.

offsetter property offsets [Function]
Apply offsets to the default values of property of grob. Offsets are restricted to immutable properties and values of type number, number-pair, or number-pair-list.
old-straight-flag grob
Old straight flag style (for composers like Bach). The angles of the flags are both 45 degrees.

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 paper-book
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 paper-book
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 paper-book
Optimally break (pages and lines) the Paper_book object paper-book to minimize badness for both vertical and horizontal spacing.

ly:option-usage port internal
Print ly:set-option usage. Optional port argument for the destination defaults to current output port. Specify internal to get doc for internal options.

ly:otf->cff otf-file-name idx
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
Is font an OpenType font?

ly:otf-font-glyph-info font glyph
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
Extract a table tag from font. Return empty string for non-existent tag.

ly:otf-glyph-count font
Return the number of glyphs in font.

ly:otf-glyph-list font
Return a list of glyph names for font.

ly:output-def? x
Is x a smob of class Output_def?

ly:output-def-clone def
Clone output definition def.

ly:output-def-lookup def sym val
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 output-def default-value
Return the parent output definition of output-def, or default-value if output-def has no parent. default-value is optional, and defaults to ().
ly:output-def-scope  def
  Return the variable scope inside def.

ly:output-def-set-variable!  def sym val
  Set an output definition def variable sym to val.

ly:output-description  output-def
  Return the description of translators in output-def.

ly:output-find-context-def  output-def context-name
  Return an alist of all context defs (matching context-name if given) in output-def.

output-module?  module
  Return #t if module belongs to an output module usually carrying context definitions (\midi or \layout).

ly:outputter-close  outputter
  Close port of outputter.

ly:outputter-dump-stencil  outputter stencil
  Dump stencil expr onto outputter.

ly:outputter-dump-string  outputter str
  Dump str onto outputter.

ly:outputter-output-scheme  outputter expr
  Output expr to the paper outputter.

ly:outputter-port  outputter
  Return output port for outputter.

oval-stencil  stencil thickness x-padding y-padding
  Add an oval around stencil, padded by the padding pair, producing a new stencil.

override-head-style  heads style
  Override style for heads to style.

override-time-signature-setting  time-signature setting
  Override the time signature settings for the context in time-signature, with the new setting alist setting.

ly:page-marker?  x
  Is x a smob of class Page_marker?

ly:page-turn-breaking  paper-book
  Optimally break (pages and lines) the Paper_book object paper-book such that page turns only happen in specified places, returning its pages.

ly:pango-font?  f
  Is f a Pango font?

ly:pango-font-physical-fonts  f
  Return alist of (ps-name file-name font-index) lists for Pango font f.

pango-pf-file-name  pango-pf
  Return the file name of the Pango physical font pango-pf.
pango-pf-font-name  
Return the font name of the Pango physical font  pango-pf.

pango-pf-fontindex  
Return the font index of the Pango physical font  pango-pf.

ly:paper-book?  x  
Is  x  a smob of class  Paper_book?

ly:paper-book-header  pb  
Return the header definition  (\header) in  Paper_book  object  pb.

ly:paper-book-pages  pb  

ly:paper-book-paper  pb  
Return the paper output definition  (\paper) in  Paper_book  object  pb.

ly:paper-book-performances  pb  

ly:paper-book-scopes  pb  

ly:paper-book-systems  pb  

ly:paper-column::break-align-width  col align-syms  
col  should be a non-musical paper-column. This function determines the horizontal extent of a break align group contained in this column, relative to the system. The break align group is searched according to  align-sym, which is either a break align symbol (see the break-align-symbol property), or a list of such symbols. For example,

(ly:paper-column::break-align-width  col '(key-signature staff-bar))

tries to find a BreakAlignGroup of key signatures, but falls back on bar lines if there are no key signatures or if the extent of the BreakAlignGroup containing them is empty (for example, if they are omitted).

The special symbol  break-alignment  means the combined extent of all items in the paper column. It is useful as the last element of the list, for a catch-all fallback.

This function never returns an empty interval. If no matching group is found or the group has an empty extent, it returns a point interval at the coordinate of the column relative to the system.

ly:paper-column::print  
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  
Return a list containing the fonts from output definition  def (e.g.,  \paper).

ly:paper-get-font def chain  
Find a font metric in output definition  def  satisfying the font qualifiers in alist  chain, and return it. (An alist chain is a list of alists, containing grob properties.)

ly:paper-get-number def sym  
Return the value of variable  sym  in output definition  def  as a double.
ly:paper-outputs__\texttt{scale} \texttt{def}
Return the output-scale for output definition \texttt{def}.

\texttt{ly:paper-score\texttt{-paper-systems}} \texttt{paper-score}
Return vector of \texttt{paper-system} objects from \texttt{paper-score}.

\texttt{ly:paper-system? obj}
Is \texttt{obj} a C++ \texttt{Prob} object of type \texttt{paper-system}?

\texttt{ly:paper-system-minimum-distance} \texttt{sys1 sys2}
Measure the minimum distance between two paper system \texttt{Probs} \texttt{sys1} and \texttt{sys2}, using their stored skylines if possible and falling back to their extents otherwise.

\texttt{parenthesize-stencil} \texttt{stencil half-thickness width angularity padding}
Add parentheses around \texttt{stencil}, returning a new stencil.

\texttt{ly:parse-file name}
Parse a single .ly file. Upon failure, throw \texttt{ly-file-failed} key.

\texttt{ly:parse-init name}
Parse the init file \texttt{name}.

\texttt{ly:parse-string-expression} \texttt{parser-smob ly-code filename line}
Parse the string \texttt{ly-code} with \texttt{parser-smob}. Return the contained music expression. \texttt{filename} and \texttt{line} are optional source indicators.

\texttt{parse-terse-string} \texttt{terse-definition}
Parse a \texttt{fret-diagram-terse} definition string \texttt{terse-definition} and return a marking list, which can be used with a fretboard grob.

\texttt{ly:parsed-undead-list!}
Return the list of objects that have been found alive but should have been dead, and clear that list.

\texttt{ly:parser-clear-error} \texttt{parser}
Clear error flag for \texttt{parser}, defaulting to current parser.

\texttt{ly:parser-clone} \texttt{closures location}
Return a clone of current parser. An association list of port positions to closures can be specified in \texttt{closures} in order to have $\$ and $\# interpreted in their original lexical environment. If \texttt{location} is a valid location, it becomes the source of all music expressions inside.

\texttt{ly:parser-define!} \texttt{symbol val}
Bind \texttt{symbol} to \texttt{val} in current parser’s module.

\texttt{ly:parser-error} \texttt{msg input}
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.

\texttt{ly:parser-has-error?} \texttt{parser}
Does \texttt{parser} (defaulting to current parser) have an error flag?

\texttt{ly:parser-include-string} \texttt{ly-code}
Include the string \texttt{ly-code} into the input stream for current parser. Can only be used in immediate Scheme expressions ($\$ instead of $\#$).

\texttt{ly:parser-lookup} \texttt{symbol}
Look up \texttt{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.

percussion? *instrument* [Function]
   Return #t if the instrument should use MIDI channel 9.

ly:performance-headers *performance* [Function]
   Return the list of headers with the innermost first.

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 smob of class Pitch?

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 pitch *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 pitch *pp* from middle C.

ly:pitch-semitones *pp* [Function]
   Calculate the number of semitones of pitch *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 pitch *pp* from middle C as a rational number.

ly:pitch-transpose *p delta* [Function]
   Transpose pitch *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.
Chapter 4: Scheme functions

polar->rectangular  \textit{radius angle-in-degrees}  \quad \textbf{[Function]}
\begin{itemize}
\item Return polar coordinates (\textit{radius}, \textit{angle-in-degrees}) as rectangular coordinates (x-length, y-length).
\end{itemize}

\textit{ly:position-on-line? sg spos}  \quad \textbf{[Function]}
\begin{itemize}
\item Return whether \textit{spos} is on a line of the staff associated with the grob \textit{sg} (even on an extender line).
\end{itemize}

\textit{prepend-alist-chain key val chain}  \quad \textbf{[Function]}
\begin{itemize}
\item Convenience to make a new alist chain from \textit{chain} by prepending a binding of \textit{key} to \textit{val}.
\item This is similar to \texttt{acons}, for alist chains (lists of alists).
\end{itemize}

\textit{ly:prob? x}  \quad \textbf{[Function]}
\begin{itemize}
\item Is \textit{x} a smob of class \texttt{Prob}?
\end{itemize}

\textit{ly:prob-immutable-properties prob}  \quad \textbf{[Function]}
\begin{itemize}
\item Retrieve an alist of immutable properties.
\end{itemize}

\textit{ly:prob-mutable-properties prob}  \quad \textbf{[Function]}
\begin{itemize}
\item Retrieve an alist of mutable properties.
\end{itemize}

\textit{ly:prob-property prob sym val}  \quad \textbf{[Function]}
\begin{itemize}
\item Return the value for property \textit{sym} of \texttt{Prob} object \textit{prob}. If no value is found, return \textit{val} or \texttt{'}() if \textit{val} is not specified.
\end{itemize}

\textit{ly:prob-property? obj sym}  \quad \textbf{[Function]}
\begin{itemize}
\item Is boolean prop \textit{sym} of \texttt{obj} set?
\end{itemize}

\textit{ly:prob-set-property! obj sym value}  \quad \textbf{[Function]}
\begin{itemize}
\item Set property \textit{sym} of \texttt{obj} to \textit{value}.
\end{itemize}

\textit{ly:prob-type? obj type}  \quad \textbf{[Function]}
\begin{itemize}
\item Is \texttt{obj} the specified \texttt{prob} type?
\end{itemize}

\textit{ly:programming-error str rest}  \quad \textbf{[Function]}
\begin{itemize}
\item A Scheme callable function to issue the internal warning \textit{str}. The message is formatted with \texttt{format}; \textit{rest} holds the formatting arguments (if any).
\end{itemize}

\textit{ly:progress str rest}  \quad \textbf{[Function]}
\begin{itemize}
\item A Scheme callable function to print progress \textit{str}. The message is formatted with \texttt{format}; \textit{rest} holds the formatting arguments (if any).
\end{itemize}

\textit{ly:property-lookup-stats sym}  \quad \textbf{[Function]}
\begin{itemize}
\item Return hash table with a property access corresponding to \textit{sym}. Choices are \texttt{prob}, \texttt{grob}, and \texttt{context}.
\end{itemize}

\textit{ly:pt num}  \quad \textbf{[Function]}
\begin{itemize}
\item \textit{num} printer points.
\end{itemize}

\textit{ly:pure-call data grob start end rest}  \quad \textbf{[Function]}
\begin{itemize}
\item Convert property \textit{data} (unpure-pure container or procedure) to value in a pure context defined by \textit{grob}, \textit{start}, \textit{end}, and possibly \textit{rest} arguments.
\end{itemize}

\textit{pure-chain-offset-callback grob start end prev-offset}  \quad \textbf{[Function]}
\begin{itemize}
\item Sometimes, a chained offset callback is unpure and there is no way to write a pure function that estimates its behavior. In this case, we use a pure equivalent that will simply pass the previous calculated offset value.
\end{itemize}
ly:randomize-rand-seed
Randomize C random generator.

ratio->fret ratio
Calculate a fret number given ratio for the harmonic.

ratio->pitch ratio
Calculate a pitch given ratio for the harmonic.

read-lily-expression chr port
Read a lilypond music expression enclosed within #{} and #} from port and return the corresponding Scheme music expression. ‘$’ and ‘#’ introduce immediate and normal Scheme forms.

recording-group-emulate music odef
Interpret music according to odef, but store all events in a chronological list, similar to the Recording_group_engraver in LilyPond version 2.8 and earlier.

ly:register-stencil-expression symbol
Add symbol as head of a stencil expression.

ly:register-translator creator name description
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
Determine the extent of elements relative to common in the axis direction.

remove-grace-property context-name grob sym
Remove all sym for grob in context-name.

remove-whitespace strg
Remove characters satisfying char-whitespace? from string strg.

ly:rename-file oldname newname
Rename oldname to newname. In contrast to Guile’s rename-file function, this replaces the destination if it already exists. On Windows, fall back to copying the file contents if newname cannot be deleted.

ly:reset-all-fonts
Forget all about previously loaded fonts.

retrieve-glyph-flag flag-style dir dir-modifier grob
Load the correct flag glyph from the font.

retrograde-music music
Return music in retrograde (reversed) order.

revert-fontSize func-name mag
Used by \magnifyMusic and \magnifyStaff. Calculate the previous fontSize value (before scaling) by factoring out the magnification factor mag (if func-name is 'magnifyMusic), or by factoring out the context property magnifyStaffValue (if func-name is 'magnifyStaff). Revert the fontSize in the appropriate context accordingly.
With \magnifyMusic, the scaling is reverted after the music block it operates on. \magnifyStaff does not operate on a music block, so the scaling from a previous call (if there is one) is reverted before the new scaling takes effect.
revert-head-style heads
  Revert style for heads.

revert-props func-name mag props
  Used by \magnifyMusic and \magnifyStaff. Revert each prop in props in the appropriate context. func-name is either 'magnifyMusic or 'magnifyStaff. The props list is formatted like:

  '((Stem thickness)
   (Slur line-thickness)
   ...)

ly:round-filled-box xext yext blot
  Make a Stencil object that prints a black box of dimensions xext, yext and roundness blot.

ly:round-polygon points blot extroversion filled-scm
  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 the default of 0 keeping the middle of the line just on the polygon.

rounded-box-stencil stencil thickness padding blot
  Add a rounded box around stencil, producing a new stencil.

ly:run-translator mus output-def
  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.

scale-beam-thickness mag
  Used by \magnifyMusic. Scaling Beam.beam-thickness exactly to the mag value will not work. This uses two reference values for beam-thickness to determine an acceptable value when scaling, then does the equivalent of a \temporary \override with the new value.

scale-fontSize func-name mag
  Used by \magnifyMusic and \magnifyStaff. Look up the current fontSize in the appropriate context and scale it by the magnification factor mag. func-name is either 'magnifyMusic or 'magnifyStaff.

scale-layout paper scale
  Return a clone of paper, scaled by the given scale factor.

scale-props func-name mag allowed-to-shrink? props
  Used by \magnifyMusic and \magnifyStaff. For each prop in props, find the current value of the requested prop, scale it by the magnification factor mag, and do the equivalent of a \temporary \override with the new value in the appropriate context. If allowed-to-shrink? is #f, don't let the new value be less than the current value. func-name is either 'magnifyMusic or 'magnifyStaff. The props list is formatted like:

  '((Stem thickness)
   (Slur line-thickness)
   ...)

ly:score? x
  Is x a smob of class Score?

ly:score-add-output-def! score def
  Add an output definition def to score.
ly:score-embedded-format score layout
   Run score through layout (an output definition) scaled to correct output-scale already, returning a list of layout lines.

ly:score-error? score
   Was there an error in the score?

ly:score-header score
   Return score header.

ly:score-music score
   Return score music.

ly:score-output-defs score
   All output definitions in a score.

ly:score-set-header! score module
   Set the score header.

scorify-music music
   Preprocess music.

seconds->moment s context
   Return a moment equivalent to s seconds at the current tempo.

select-head-glyph style log
   Select a note head glyph string based on note head style style and duration log log.

self-alignment-interface::self-aligned-on-breakable grob
   Return the X-offset that places grob according to its self-alignment-X over the reference point defined by the break-align-anchor-alignment of a break-aligned item such as a Clef.

sequential-music-to-chord-exceptions seq rest ...
   Transform sequential music seq of type
      \markup{ foobar }
   to (cons cde-pitches foobar-markup), or to (cons de-pitches foobar-markup) if omit-root is given and non-false.

set-accidental-style style rest ...
   Set accidental style to style. Optionally take a context argument, e.g., 'Staff or 'Voice. The context defaults to Staff, except for piano styles, which use GrandStaff as a context.

ly:set-color-names alist
   Define named colors for ly:stencil-in-color. alist has the entries of the format (name . color), where color is a list of length 3 (RGB) or 4 (RGB+alpha).

ly:set-default-scale scale
   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.

set-global-staff-size sz
   Set the default staff size, where sz is thought to be in points.
ly:set-grob-creation-callback cb
Specify a procedure that gets called every time a new grob is created. The callback receives as arguments the grob that was created, the name of the C++ source file that caused the grob to be created, and the corresponding line number in the C++ source file. Call with #f as argument to unset the callback.

ly:set-grob-modification-callback cb
Specify a procedure that gets called every time LilyPond modifies a grob property. The callback receives 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. Call with #f as argument to unset the callback.

ly:set-middle-C! context
Set the middleCPosition variable in context based on the variables middleCClefPosition and middleCOffset.

set-mus-properties! m alist
Set all of alist as properties of m.

ly:set-option var val
Set a program option.

ly:set-origin! m origin
Set the origin given in origin to m. m is typically 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.

set-output-property grob-name symbol val
Usage example: \(\text{\#app\#output \#(set-output-property 'Clef 'extra-offset '(0 . 1))}\)

ly:set-property-cache-callback cb
Specify a procedure that gets called whenever LilyPond calculates a callback function and caches the result. The callback receives 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. Call with #f as argument to unset the callback.

shift-one-duration-log music shift dot
Add shift to duration-log of 'duration in music and optionally dot to any note encountered. The number of dots in the shifted music may not be less than zero.

shift-right-at-line-begin g
Shift an item to the right, but only at the start of the line.

shift-semitone->pitch key semitone->pitch
Given a function semitone->pitch converting a semitone number into a note value for a lookup table created in relation to C, returns a corresponding function in relation to key. The note values returned by this function differ only enharmonically from the original semitone->pitch function.
skip->rest mus
Replace mus by RestEvent of the same duration if it is a SkipEvent. Useful for extracting parts from crowded scores.

skip-of-length mus
Create a skip of exactly the same length as mus.

skip-of-moment-span start-moment end-moment
Make skip music fitting between start-moment and end-moment. The grace part of end-moment matters only if start-moment and end-moment have the same main part.

ly:skyline? x
Is x a smob of class Skyline?

ly:skyline-distance skyline other-skyline horizon-padding
Compute the distance between the two skylines, padding by horizon-padding if provided.

ly:skyline-empty? sky
Return whether skyline sky is empty.

ly:skyline-height skyline x
Return the height of skyline at point x.

ly:skyline-max-height skyline
Return the maximum height found in skyline.

ly:skyline-max-height-position skyline
Return the position at which skyline reaches its maximum height.

ly:skyline-pad skyline horizon-padding
Return a version of skyline padded by horizon-padding along the horizon.

ly:skyline-touching-point skyline other-skyline horizon-padding
Get the point where skyline and other-skyline (having opposite directions) reach their minimum distance. If horizon-padding is provided, one skyline is padded with it first.

ly:skylines-for-stencil stencil axis
Return a pair of skylines representing the outline of stencil. This gives horizontal skylines if axis is X, and vertical skylines with axis = Y.

ly:smob-protects
Return LilyPond’s internal smob protection list.

ly:solve-spring-rod-problem springs rods length ragged
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
Is x a smob of class Source_file?
ly:source-files parser-smob
Return a list of input files that have been opened up to here, including the files that have been closed already. A parser, parser-smob, may optionally be specified.

ly:span-bar::before-line-breaking grob
A dummy callback that kills the Grob grob if it contains no elements.

ly:span-bar::calc-anchor grob
Calculate the anchor position of the SpanBar. The anchor is used for the correct placement of bar numbers, etc.

ly:span-bar::calc-glyph-name grob
Return the 'glyph-name of the corresponding BarLine grob. The corresponding SpanBar glyph is computed within span-bar::compound-bar-line.

span-bar::compound-bar-line grob bar-glyph extent
Build the stencil of the span bar.

ly:span-bar::print grob
The print routine for span bars.

ly:span-bar::width grob
Compute the width of the SpanBar stencil.

Span_stem_engraver ctx
Connect cross-staff stems to the stems above in the system.

ly:spanner? g
Is g a spanner object?

ly:spanner-bound spanner dir def
Get one of the bounds of spanner. dir is -1 for left, and 1 for right. If the spanner does not (yet) have a bound for this direction, return def, or '()' if def is not specified.

ly:spanner-broken-into spanner
Return broken-into list for spanner.

ly:spanner-set-bound! spanner dir item
Set grob item as bound in direction dir for spanner.

ly:spawn command rest
Simple Scheme interface to the GLib function g_spawn_sync. If an error occurs, format it with format and rest.

split-list-by-separator lst pred
Split lst at each element that satisfies pred, and return the parts (with the separators removed) as a list of lists. Example:

(split-list-by-separator '(a 0 b c 1 d) number?)
⇒ ((a) (b c) (d))

ly:spring? x
Is x a smob of class Spring?

ly:spring-set-inverse-compress-strength! spring strength
Set the inverse compress strength of spring.

ly:spring-set-inverse-stretch-strength! spring strength
Set the inverse stretch strength of spring.
stack-lines dir padding baseline stils
   Stack stencils vertically with a baseline skip.

stack-stencil-line space stencils
   Adjoin a list of stencils along the x axis, leaving space between the end of each stencil and the beginning of the following stencil. Stencils with empty y extent are not given space before them and don't avoid overlapping other stencils.

stack-stencils axis dir padding stils
   Stack stencils stils in direction axis, dir, using padding.

stack-stencils-padding-list axis dir paddings stils
   Stack stencils stils in direction axis, dir, using a list of paddings.

staff-ellipsis::calc-y-extent grob
   Callback for StaffEllipsis grob, which is used with skipTypesetting.

staff-ellipsis::print grob
   Callback for StaffEllipsis grob, which is used with skipTypesetting.

ly:staff-symbol-line-thickness grob
   Return 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
   Return the radius of the staff associated with grob.

ly:staff-symbol-staff-space grob
   Return 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:stderr-redirect fd-or-file-name mode
   Redirect standard error output (stderr) to file descriptor fd if the first parameter is an integer, or to file file-name, opened with mode.

ly:stencil? x
   Is x a smob of class Stencil?

ly:stencil-add args
   Combine stencils. Takes any number of arguments.

ly:stencil-aligned-to stil axis dir
   Align stencil 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
   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
   Return whether stil is empty. If an optional axis is supplied, the emptiness check is restricted to that axis.

ly:stencil-expr stil
   Return the expression of stencil stil.
ly:stencil-extent stencil axis
Return a pair of numbers signifying the extent of stencil stencil in axis direction (0 or 1 for x and y axis, respectively).

ly:stencil-in-color stc r g b a
Put stencil stc in a different color. Accepts either three values for r, g, b and an optional value for a, or a single CSS-like string.

ly:stencil-outline stencil outline
Return a stencil with the stencil expression (inking) of stencil stencil but with outline and dimensions from stencil outline.

ly:stencil-rotate stencil angle x y
Return a stencil stencil rotated by angle degrees around the relative offset (x, y). E.g., an offset of (-1, 1) rotates the stencil around the left upper corner.

ly:stencil-rotate-absolute stencil angle x y
Return a stencil stencil rotated by angle degrees around point (x, y), given in absolute coordinates.

ly:stencil-scale stencil x y
Scale stencil stencil using the horizontal and vertical scaling factors x and optional y (defaulting to x). Negative values flip or mirror stencil without changing its origin; this may result in collisions unless it is repositioned.

ly:stencil-stack first axis direction second padding mindist
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 stencil offset
Return a copy of stencil stencil but translated by offset (a pair of numbers).

ly:stencil-translate-axis stencil amount axis
Return a copy of stencil stencil but translated by amount in axis direction.

stencil-true-extent stencil axis
Return the extent of the actual printed ink of stencil on axis.

stencil-whiteout stencil [style [thickness [line-thickness]]]
White-out a stencil (i.e., add a white background around it).
style, thickness and line-thickness are optional arguments. If set, style determines the shape of the white background. Given 'outline the white background is produced by stencil-whiteout-outline, given 'rounded-box it is produced by stencil-whiteout-box with rounded corners, given other arguments (e.g., 'box) or when unspecified it defaults to stencil-whiteout-box with square corners. If thickness is specified it determines how far, as a multiple of line-thickness, the white background extends past the extents of stencil. If thickness has not been specified, an appropriate default is chosen based on style.

stencil-whiteout-box stencil [thickness [blot [color]]]
White-out a stencil by printing it on top of a white (or color) rectangle.

thickness is how far, as a multiple of line-thickness, the white outline extends past the extents of stencil stencil.
**stencil-whiteout-outline**  
*stil [thickness [color [angle-increments [[radial-increments]]]]]

White-out a stencil by surrounding it with white (or color) around its outline.

This function works by creating a series of white or color stencils radially offset from the original stencil with angles from 0 to 2\pi, at an increment of angle-inc, and with radii from radial-inc to thickness. thickness is how big the white outline is, as a multiple of line-thickness. radial-increments is how many copies of the white stencil we make on our way out to thickness. angle-increments is how many copies of the white stencil we make between 0 and 2\pi.

**straight-flag**  
*flag-thickness flag-spacing upflag-angle upflag-length downflag-angle downflag-length*

Create a stencil for a straight flag. flag-thickness and flag-spacing are given in staff spaces, upflag-angle and downflag-angle are given in degrees, and upflag-length and downflag-length are given in staff spaces.

All lengths are scaled according to the font size of the note.

**ly:stream-event?**  
*obj*

Is obj a Stream_event object?

**ly:string-percent-encode**  
*str*

Encode all characters in string str with hexadecimal percent escape sequences, with the following exceptions: characters ‘-./_’ and characters in ranges 0-9, A-Z, and a-z.

**ly:string-substitute**  
*a b s*

Replace string a by string b in string s.

**style-note-heads**  
*heads style music*

Set style for all heads in music. Works both inside of and outside of chord construct.

**suggest-convert-ly-message**  
*version-seen*

Internally used when the file has an error, to suggest usage of convert-ly if the \version statement is considered outdated compared to the LilyPond version that is running.

**symbol-concatenate**  
*names . . .*

Like string-concatenate, but for symbols.

**ly:system-font-load**  
*name*

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.

**tag-group-get**  
*tag*

Return the tag group (as a list of symbols) that the given tag symbol belongs to, #f if none.

**tags-keep-predicate**  
*tags*

Return a predicate that returns #f for any music that is to be removed by \keepWithTag on the given symbol or list of symbols tags.

**tags-remove-predicate**  
*tags*

Return a predicate that returns #f for any music that is to be removed by \removeWithTag on the given symbol or list of symbols tags.
teaching-accidental-rule  context pitch barnum  
An accidental rule that typesets a cautionary accidental if it is included in the key signature and does not directly follow a note on the same staff line.

ly:text-interface::interpret-markup  
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:time-signature::print grob  
Print routine for time signatures.

ly:transform? x  
Is x a smob of class Transform?

ly:transform->list transform  
Convert a transform matrix to a list of six values. Values are xx, yx, xy, yy, x0, y0.

ly:translate-cpp-warning-scheme str  
Translate a string in C++ printf format and modify it to use it for Scheme formatting.

ly:translator? x  
Is x a smob of class Translator?

ly:translator-context trans  
Return the context of the translator object trans.

ly:translator-description creator  
Return an alist of properties of translator definition creator.

ly:translator-group? x  
Is x a smob of class Translator_group?

ly:translator-name creator  
Return the type name of the translator definition creator. The name is a symbol.

ly:transpose-key-alist l pit  
Make a new key alist of l transposed by pitch pit.

ly:ttf->pfa ttf-file-name idx  
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  
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  
Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in 
PFA format, pass it through.

unbroken-or-first-broken-spanner? spanner  
Is spanner either unbroken or the first of its broken siblings?
unbroken-or-last-broken-spanner? spanner
Is spanner either unbroken or the last of its broken siblings?

unbroken-spanner? spanner
Is spanner unbroken? A spanner has to be broken if it spans more than one system, or if one of its bounds is on the limit of the system. This function returns #f on the clones, but #t on the originals.

unfold-repeats types music
Replace repeats of the types given by types with unfolded repeats. If types is an empty list, repeated-music is taken, unfolding all.

unfold-repeats-fully music
Unfold repeats and expand the resulting unfolded-repeated-music.

uniq-list lst
Remove doublets from list lst (i.e., make its elements unique), assuming that it is sorted. Uses equal? for comparisons.

ly:unit
Return the unit used for lengths as a string.

unity-if-multimeasure context dur
Given a context and a duration, return 1 if the duration is longer than the measureLength in that context, and #f otherwise. This supports historic use of Completion_heads_engraver to split c1*3 into three whole notes.

ly:unpure-call data grob rest
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
Is x a smob of class Unpure_pure_container?

ly:unpure-pure-container-pure-part pc
Return the pure part of pc.

ly:unpure-pure-container-unpure-part pc
Return the unpure part of pc.

ly:usage
Print usage message.

value-for-spanner-piece property args
Associate a piece of broken spanner grob with an element of list arg.

ly:verbose-output?
Was verbose output requested, i.e., is the log level at least DEBUG?

ly:version
Return the current LilyPond version as a list, e.g., (1 3 127 uu1).

ly:version? op ver
Use operator op to compare the currently executed LilyPond version with a given version ver, which is passed as a list of numbers.
### voicify-music

\texttt{voicify-music \textit{m [id]}}

Recursively split chords that are separated with `\`. Optional \textit{id} can be a list of context ids to use. If numeric, they also indicate a voice type override. If \textit{id} is just a single number, that’s where numbering starts.

### volta-bracket::calc-hook-visibility

\texttt{volta-bracket::calc-hook-visibility \textit{bar-glyph}}

Determine the visibility of the volta bracket end hook, returning \#t if \textit{no} hook should be drawn.

### ly:volta-bracket::calc-shorten-pair

\texttt{ly:volta-bracket::calc-shorten-pair \textit{grob}}

Calculate the shorten-pair values for an ideal placement of the volta brackets relative to the bar lines.

### volta-spec-music

\texttt{volta-spec-music \textit{number-list music}}

Add `\ volta number-list to \textit{music}.`

### ly:warning

\texttt{ly:warning \textit{str rest}}

A Scheme callable function to issue the warning \textit{str}. The message is formatted with \texttt{format}; \textit{rest} holds the formatting arguments (if any).

### ly:warning-located

\texttt{ly:warning-located \textit{location str rest}}

A Scheme callable function to issue the warning \textit{str} at the specified location in an input file. The message is formatted with \texttt{format}; \textit{rest} holds the formatting arguments (if any).

### ly:wide-char->utf-8

\texttt{ly:wide-char->utf-8 \textit{wc}}

Encode the Unicode codepoint \textit{wc}, an integer, as UTF-8.

### write-me

\texttt{write-me \textit{message x}}

Return \textit{x}. Display \textit{message} and write \textit{x}. Handy for debugging, possibly turned off.
Appendix A Indices

A.1 Concept index

(Index is nonexistent)

A.2 Function index

A

add-bar-glyph-print-procedure 794
add-grace-property 794
add-music-fonts 794
add-new-clef 794
add-simple-time-signature-style 794
add-stroke-glyph 794
add-stroke-straight 794
alist->hash-table 795
allow-volta-hook 795
alterations-in-key 795
angle-0-2pi 795
angle-0-360 795
arrow-stencil 795
arrow-stencil-maker 795
assoc-get 795
at-bar-line-substitute-caesura-type 796

B

bar-line::calc-break-visibility 796
bar-line::calc-glyph-name 796
bar-line::calc-glyph-name-for-direction 796
bar-line::compound-bar-line 796
bar-line::draw-filled-box 796
bar-line::widen-bar-extent-on-span 796
base-length 796
beam-exceptions 796
beat-structure 796
bend-spanner::print 797
bend::arrow-head-stencil 796
bend::calc-bend-x-begin 797
bend::calc-bend-x-end 797
bend::target-cautionary 797
bend::text-string 797
book-first-page 797
box-grob-stencil 798
box-stencil 798
bracketify-stencil 798
break-alignable-interface::
  self-alignment-of-anchor 798
break-alignable-interface::
  self-alignment-opposite-of-anchor 798
break-alignment-list 798

C

caesura-script-interface::
  before-line-breaking 798
caesura-to-bar-line-or-divisio 799
caesura-to-divisio 799
calc-harmonic-pitch 799
centered-spanner-interface::
  calc-x-offset 799
centered-stencil 799
chain-assoc-get 799
change-pitches 799
check-context-path 799
check-grob-path 800
check-music-path 800
chord-name->german-markup 800
chord-name->italian-markup 800
circle-stencil 800
clef-transposition-markup 800
collect-book-music-for-book 800
collect-bookpart-for-book 800
collect-music-aux 800
collect-music-for-book 800
comparator-from-key 800
construct-chord-elements 801
context-spec-music 802
copy-repeat-chord 802
count-list 802
create-glyph-flag 802
cross-staff-connect 802
cue-substitute 802
cyclic-base-value 802

default-flag 802
define-bar-line 803
define-event-class 803
define-event-function 803
define-fonts 803
define-markup-command 803
define-markup-list-command 804
define-music-function 804
define-scheme-function 804
define-syntax-function 804
define-tag-group 804
define-void-function 805
degrees->radians 805
descend-to-context 805
determine-split-list 805
determine-string-fret-finger 805
dir-basename 805
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ly:input-file-line-char-column</td>
<td>816</td>
</tr>
<tr>
<td>ly:input-location?</td>
<td>816</td>
</tr>
<tr>
<td>ly:input-message</td>
<td>816</td>
</tr>
<tr>
<td>ly:input-warning</td>
<td>816</td>
</tr>
<tr>
<td>ly:interpret-music-expression</td>
<td>817</td>
</tr>
<tr>
<td>ly:intlog2</td>
<td>817</td>
</tr>
<tr>
<td>ly:item-break-dir</td>
<td>817</td>
</tr>
<tr>
<td>ly:item-get-column</td>
<td>817</td>
</tr>
<tr>
<td>ly:item?</td>
<td>817</td>
</tr>
<tr>
<td>ly:iterator?</td>
<td>817</td>
</tr>
<tr>
<td>ly:length</td>
<td>817</td>
</tr>
<tr>
<td>ly:lily-lexer?</td>
<td>817</td>
</tr>
<tr>
<td>ly:lily-parser?</td>
<td>818</td>
</tr>
<tr>
<td>ly:line-interface::line</td>
<td>818</td>
</tr>
<tr>
<td>ly:listened-event-class?</td>
<td>818</td>
</tr>
<tr>
<td>ly:listened-event-types?</td>
<td>818</td>
</tr>
<tr>
<td>ly:listener?</td>
<td>818</td>
</tr>
<tr>
<td>ly:make-book</td>
<td>819</td>
</tr>
<tr>
<td>ly:make-book-part</td>
<td>819</td>
</tr>
<tr>
<td>ly:make-context-mod</td>
<td>819</td>
</tr>
<tr>
<td>ly:make-dispatcher</td>
<td>819</td>
</tr>
<tr>
<td>ly:make-duration</td>
<td>819</td>
</tr>
<tr>
<td>ly:make-global-context</td>
<td>820</td>
</tr>
<tr>
<td>ly:make-global-translator</td>
<td>820</td>
</tr>
<tr>
<td>ly:make-grob-properties</td>
<td>820</td>
</tr>
<tr>
<td>ly:make-liner</td>
<td>820</td>
</tr>
<tr>
<td>ly:make-moment</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-music</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-music-function</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-music-relative!</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-output-def</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-page-label-marker</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-page-permission-marker</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-pango-description-string</td>
<td>821</td>
</tr>
<tr>
<td>ly:make-paper-outputter</td>
<td>822</td>
</tr>
<tr>
<td>ly:make-pitch</td>
<td>822</td>
</tr>
<tr>
<td>ly:make-prob</td>
<td>822</td>
</tr>
<tr>
<td>ly:make-rotation</td>
<td>823</td>
</tr>
<tr>
<td>ly:make-scaling</td>
<td>823</td>
</tr>
<tr>
<td>ly:make-score</td>
<td>823</td>
</tr>
<tr>
<td>ly:make-spring</td>
<td>824</td>
</tr>
<tr>
<td>ly:make-stencil</td>
<td>824</td>
</tr>
<tr>
<td>ly:make-stream-event</td>
<td>824</td>
</tr>
<tr>
<td>ly:make-transform</td>
<td>824</td>
</tr>
<tr>
<td>ly:make-translation</td>
<td>824</td>
</tr>
<tr>
<td>ly:make-translation-context</td>
<td>824</td>
</tr>
<tr>
<td>ly:message</td>
<td>827</td>
</tr>
<tr>
<td>ly:minimal-breaking</td>
<td>827</td>
</tr>
<tr>
<td>ly:mm</td>
<td>827</td>
</tr>
<tr>
<td>ly:module-&gt;alist</td>
<td>827</td>
</tr>
<tr>
<td>ly:modules-&gt;lookup</td>
<td>827</td>
</tr>
<tr>
<td>ly:modules-lookup</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-add</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-div</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-grace</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-grace-denominator</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-grace-numerator</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-main</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-main-denominator</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-main-numerator</td>
<td>827</td>
</tr>
<tr>
<td>ly:moment-mod</td>
<td>828</td>
</tr>
<tr>
<td>ly:moment-mul</td>
<td>828</td>
</tr>
<tr>
<td>ly:moment-sub</td>
<td>828</td>
</tr>
<tr>
<td>ly:moment?</td>
<td>827</td>
</tr>
</tbody>
</table>
### R
- ratio->fret ........................................... 837
- ratio->pitch ....................................... 837
- read_lily_expression ............................... 837
- recording-group-emulate .......................... 837
- remove-grace-property ............................. 837
- remove-whitespace ................................ 837
- retrieve-glyph-flag ............................... 837
- retrograde-music ................................ 837
- revert-font-size .................................. 837
- revert-head-style .................................. 838
- revert-props ...................................... 838
- rounded-box-stencil .............................. 838

### S
- scale-beam-thickness ............................. 838
- scale-fontSize .................................... 838
- scale-layout ...................................... 838
- scale-props ...................................... 838
- scorify-music .................................... 839
- seconds->moment .................................. 839
- select-head-glyph ................................ 839
- self-alignment-interface: ........................
  - self-aligned-on-breakable ...................... 839
- sequential-music-to-chord-exceptions ........... 839
- set-accidental-style .............................. 839
- set-global-staff-size ............................. 839
- set-mus-properties! ............................... 840
- set-output-property ............................... 840
- shift-one-duration-log ........................... 840
- shift-right-at-line-begin ....................... 840
- shift-semitone->pitch ............................ 840
- skip->rest ........................................ 841
- skip-of-length ................................... 841
- skip-of-moment-span .............................. 841
- span-bar::compound-bar-line ..................... 842
- Span_stem_engraver ................................ 842
- split-list-by-separator .......................... 842
- stack-lines ....................................... 843
- stack-stencil-line ................................ 843

### T
- tag-group-get ...................................... 845
- tags-keep-predicate .............................. 845
- tags-remove-predicate ............................ 845
- teaching-accidental-rule ....................... 846

### U
- unbroken-or-first-broken-spanner? .............. 846
- unbroken-or-last-broken-spanner? ............... 847
- unbroken-spanner? ................................ 847
- unfold-repeats .................................... 847
- unfold-repeats-fully ............................. 847
- uniq-list .......................................... 847
- unity-if-multimeasure ............................ 847

### V
- value-for-spanner-piece .......................... 847
- voicify-music ...................................... 848
- volta-bracket::calc-hook-visibility .......... 848
- volta-spec-music .................................. 848

### W
- write-me ........................................... 848