

LilyPond

The music typesetter

Internals Reference

The LilyPond development team

This manual is a technical reference for all internal elements used by LilyPond version 2.25.25 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 © 2000–2023 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 2.25.25

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	2
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	BarCheckEvent	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	InitialContextMusic	15
1.1.43	KeyChangeEvent	16
1.1.44	LabelEvent	16
1.1.45	LaissezVibrerEvent	17
1.1.46	LigatureEvent	17

1.1.47	LineBreakEvent.....	17
1.1.48	LyricCombineMusic.....	18
1.1.49	LyricEvent.....	18
1.1.50	MeasureCounterEvent.....	19
1.1.51	MeasureSpannerEvent.....	19
1.1.52	MultiMeasureArticulationEvent.....	19
1.1.53	MultiMeasureRestEvent.....	20
1.1.54	MultiMeasureRestMusic.....	20
1.1.55	MultiMeasureTextEvent.....	20
1.1.56	Music.....	21
1.1.57	NoteEvent.....	21
1.1.58	NoteGroupingEvent.....	22
1.1.59	OptionalMaterialEvent.....	22
1.1.60	OttavaEvent.....	22
1.1.61	OverrideProperty.....	23
1.1.62	PageBreakEvent.....	23
1.1.63	PageTurnEvent.....	23
1.1.64	PartCombineEvent.....	24
1.1.65	PartCombineMusic.....	24
1.1.66	PartialEvent.....	24
1.1.67	PartialSet.....	25
1.1.68	PercentEvent.....	25
1.1.69	PercentRepeatedMusic.....	25
1.1.70	PesOrFlexaEvent.....	26
1.1.71	PhrasingSlurEvent.....	26
1.1.72	PostEvents.....	27
1.1.73	PropertySet.....	27
1.1.74	PropertyUnset.....	27
1.1.75	QuoteMusic.....	28
1.1.76	RehearsalMarkEvent.....	28
1.1.77	RelativeOctaveCheck.....	29
1.1.78	RelativeOctaveMusic.....	29
1.1.79	RepeatSlashEvent.....	29
1.1.80	RepeatTieEvent.....	30
1.1.81	RestEvent.....	30
1.1.82	RevertProperty.....	31
1.1.83	ScriptEvent.....	31
1.1.84	SectionEvent.....	31
1.1.85	SectionLabelEvent.....	32
1.1.86	SegnoMarkEvent.....	32
1.1.87	SegnoRepeatedMusic.....	32
1.1.88	SequentialAlternativeMusic.....	33
1.1.89	SequentialMusic.....	33
1.1.90	SimultaneousMusic.....	34
1.1.91	SkipEvent.....	35
1.1.92	SkipMusic.....	35
1.1.93	SkippedMusic.....	35
1.1.94	SlurEvent.....	36
1.1.95	SostenutoEvent.....	36
1.1.96	SpacingSectionEvent.....	37
1.1.97	SpanEvent.....	37
1.1.98	StaffHighlightEvent.....	37

1.1.99	StaffSpanEvent.....	38
1.1.100	StringNumberEvent.....	38
1.1.101	StrokeFingerEvent.....	38
1.1.102	SustainEvent.....	39
1.1.103	TempoChangeEvent.....	39
1.1.104	TextMarkEvent.....	39
1.1.105	TextScriptEvent.....	40
1.1.106	TextSpanEvent.....	40
1.1.107	TieEvent.....	40
1.1.108	TimeScaledMusic.....	41
1.1.109	TimeSignatureEvent.....	41
1.1.110	TimeSignatureMusic.....	41
1.1.111	TransposedMusic.....	42
1.1.112	TremoloEvent.....	42
1.1.113	TremoloRepeatedMusic.....	43
1.1.114	TremoloSpanEvent.....	43
1.1.115	TrillSpanEvent.....	44
1.1.116	TupletSpanEvent.....	44
1.1.117	UnaCordaEvent.....	44
1.1.118	UnfoldedRepeatedMusic.....	45
1.1.119	UnfoldedSpeccedMusic.....	45
1.1.120	UnrelativableMusic.....	46
1.1.121	VoiceSeparator.....	46
1.1.122	VoltaRepeatEndEvent.....	46
1.1.123	VoltaRepeatStartEvent.....	47
1.1.124	VoltaRepeatedMusic.....	47
1.1.125	VoltaSpanEvent.....	48
1.1.126	VoltaSpeccedMusic.....	48
1.1.127	VowelTransitionEvent.....	49
1.2	Music classes.....	49
1.2.1	absolute-dynamic-event.....	49
1.2.2	ad-hoc-jump-event.....	49
1.2.3	ad-hoc-mark-event.....	49
1.2.4	alternative-event.....	49
1.2.5	annotate-output-event.....	49
1.2.6	apply-output-event.....	49
1.2.7	arpeggio-event.....	49
1.2.8	articulation-event.....	50
1.2.9	bar-check-event.....	50
1.2.10	bar-event.....	50
1.2.11	bass-figure-event.....	50
1.2.12	beam-event.....	50
1.2.13	beam-forbid-event.....	50
1.2.14	bend-after-event.....	50
1.2.15	bend-span-event.....	50
1.2.16	break-dynamic-span-event.....	50
1.2.17	break-event.....	50
1.2.18	break-span-event.....	51
1.2.19	breathing-event.....	51
1.2.20	caesura-event.....	51
1.2.21	cluster-note-event.....	51
1.2.22	coda-mark-event.....	51

1.2.23	completize-extender-event.....	51
1.2.24	crescendo-event.....	51
1.2.25	dal-segno-event.....	51
1.2.26	decrescendo-event.....	51
1.2.27	double-percent-event.....	51
1.2.28	duration-line-event.....	52
1.2.29	dynamic-event.....	52
1.2.30	episema-event.....	52
1.2.31	extender-event.....	52
1.2.32	fine-event.....	52
1.2.33	finger-glide-event.....	52
1.2.34	fingering-event.....	52
1.2.35	footnote-event.....	52
1.2.36	general-rest-event.....	52
1.2.37	glissando-event.....	52
1.2.38	harmonic-event.....	52
1.2.39	hyphen-event.....	53
1.2.40	key-change-event.....	53
1.2.41	label-event.....	53
1.2.42	laissez-vibrer-event.....	53
1.2.43	layout-instruction-event.....	53
1.2.44	ligature-event.....	53
1.2.45	line-break-event.....	53
1.2.46	lyric-event.....	53
1.2.47	mark-event.....	53
1.2.48	measure-counter-event.....	53
1.2.49	measure-spanner-event.....	54
1.2.50	melodic-event.....	54
1.2.51	multi-measure-articulation-event.....	54
1.2.52	multi-measure-rest-event.....	54
1.2.53	multi-measure-text-event.....	54
1.2.54	music-event.....	54
1.2.55	note-event.....	55
1.2.56	note-grouping-event.....	55
1.2.57	optional-material-event.....	55
1.2.58	ottava-event.....	55
1.2.59	page-break-event.....	55
1.2.60	page-turn-event.....	55
1.2.61	part-combine-event.....	55
1.2.62	partial-event.....	55
1.2.63	pedal-event.....	56
1.2.64	percent-event.....	56
1.2.65	pes-or-flexa-event.....	56
1.2.66	phrasing-slur-event.....	56
1.2.67	rehearsal-mark-event.....	56
1.2.68	repeat-slash-event.....	56
1.2.69	repeat-tie-event.....	56
1.2.70	rest-event.....	56
1.2.71	rhythmic-event.....	56
1.2.72	script-event.....	56
1.2.73	section-event.....	57
1.2.74	section-label-event.....	57

1.2.75	segno-mark-event	57
1.2.76	skip-event	57
1.2.77	slur-event	57
1.2.78	sostenuto-event	57
1.2.79	spacing-section-event	57
1.2.80	span-dynamic-event	57
1.2.81	span-event	57
1.2.82	staff-highlight-event	58
1.2.83	staff-span-event	58
1.2.84	StreamEvent	58
1.2.85	string-number-event	58
1.2.86	stroke-finger-event	58
1.2.87	structural-event	59
1.2.88	sustain-event	59
1.2.89	tempo-change-event	59
1.2.90	text-mark-event	59
1.2.91	text-script-event	59
1.2.92	text-span-event	59
1.2.93	tie-event	59
1.2.94	time-signature-event	59
1.2.95	tremolo-event	59
1.2.96	tremolo-span-event	59
1.2.97	trill-span-event	60
1.2.98	tuplet-span-event	60
1.2.99	una-corda-event	60
1.2.100	volta-repeat-end-event	60
1.2.101	volta-repeat-start-event	60
1.2.102	volta-span-event	60
1.2.103	vowel-transition-event	60
1.3	Music properties	60

2 Translation 67

2.1	Contexts	67
2.1.1	ChoirStaff	67
2.1.2	ChordGrid	69
2.1.3	ChordGridScore	74
2.1.4	ChordNames	98
2.1.5	CueVoice	100
2.1.6	Devnull	110
2.1.7	DrumStaff	111
2.1.8	DrumVoice	120
2.1.9	Dynamics	130
2.1.10	FiguredBass	135
2.1.11	FretBoards	136
2.1.12	Global	139
2.1.13	GrandStaff	139
2.1.14	GregorianTranscriptionLyrics	141
2.1.15	GregorianTranscriptionStaff	144
2.1.16	GregorianTranscriptionVoice	157
2.1.17	InternalGregorianStaff	167
2.1.18	InternalMensuralStaff	180

2.1.19	KievanStaff	194
2.1.20	KievanVoice	207
2.1.21	Lyrics	218
2.1.22	MensuralStaff	221
2.1.23	MensuralVoice	235
2.1.24	NoteNames	245
2.1.25	NullVoice	247
2.1.26	OneStaff	249
2.1.27	PetrucchiStaff	250
2.1.28	PetrucchiVoice	265
2.1.29	PianoStaff	275
2.1.30	RhythmicStaff	277
2.1.31	Score	283
2.1.32	Staff	309
2.1.33	StaffGroup	322
2.1.34	StandaloneRhythmScore	323
2.1.35	StandaloneRhythmStaff	349
2.1.36	StandaloneRhythmVoice	354
2.1.37	TabStaff	365
2.1.38	TabVoice	377
2.1.39	VaticanaLyrics	388
2.1.40	VaticanaScore	390
2.1.41	VaticanaStaff	414
2.1.42	VaticanaVoice	429
2.1.43	Voice	439
2.2	Engravers and Performers	449
2.2.1	Accidental_engraver	450
2.2.2	Alteration_glyph_engraver	451
2.2.3	Ambitus_engraver	451
2.2.4	Arpeggio_engraver	452
2.2.5	Auto_beam_engraver	452
2.2.6	Axis_group_engraver	453
2.2.7	Balloon_engraver	453
2.2.8	Bar_engraver	453
2.2.9	Bar_number_engraver	456
2.2.10	Beam_collision_engraver	457
2.2.11	Beam_engraver	457
2.2.12	Beam_performer	458
2.2.13	Beat_engraver	458
2.2.14	Beat_performer	458
2.2.15	Bend_engraver	459
2.2.16	Bend_spanner_engraver	459
2.2.17	Break_align_engraver	460
2.2.18	Breathing_sign_engraver	460
2.2.19	Caesura_engraver	460
2.2.20	Centered_bar_number_align_engraver	461
2.2.21	Chord_name_engraver	461
2.2.22	Chord_square_engraver	462
2.2.23	Chord_tremolo_engraver	462
2.2.24	Clef_engraver	462
2.2.25	Cluster_spanner_engraver	463
2.2.26	Collision_engraver	463

2.2.27	Completion_heads_engraver	463
2.2.28	Completion_rest_engraver	464
2.2.29	Concurrent_hairpin_engraver	465
2.2.30	Control_track_performer	465
2.2.31	Cue_clef_engraver	465
2.2.32	Current_chord_text_engraver	466
2.2.33	Custos_engraver	466
2.2.34	Divisio_engraver	467
2.2.35	Dot_column_engraver	467
2.2.36	Dots_engraver	468
2.2.37	Double_percent_repeat_engraver	468
2.2.38	Drum_note_performer	468
2.2.39	Drum_notes_engraver	468
2.2.40	Duration_line_engraver	469
2.2.41	Dynamic_align_engraver	469
2.2.42	Dynamic_engraver	470
2.2.43	Dynamic_performer	470
2.2.44	Episema_engraver	471
2.2.45	Extender_engraver	471
2.2.46	Figured_bass_engraver	471
2.2.47	Figured_bass_position_engraver	472
2.2.48	Finger_glide_engraver	472
2.2.49	Fingering_column_engraver	472
2.2.50	Fingering_engraver	472
2.2.51	Font_size_engraver	472
2.2.52	Footnote_engraver	473
2.2.53	Forbid_line_break_engraver	473
2.2.54	Fretboard_engraver	473
2.2.55	Glissando_engraver	474
2.2.56	Grace_auto_beam_engraver	474
2.2.57	Grace_beam_engraver	475
2.2.58	Grace_engraver	475
2.2.59	Grace_spacing_engraver	475
2.2.60	Grid_chord_name_engraver	476
2.2.61	Grid_line_span_engraver	476
2.2.62	Grid_point_engraver	476
2.2.63	Grob_pq_engraver	476
2.2.64	Horizontal_bracket_engraver	477
2.2.65	Horizontal_script_engraver	477
2.2.66	Hyphen_engraver	477
2.2.67	Instrument_name_engraver	477
2.2.68	Instrument_switch_engraver	478
2.2.69	Jump_engraver	478
2.2.70	Keep_alive_together_engraver	479
2.2.71	Key_engraver	479
2.2.72	Key_performer	480
2.2.73	Kievan_ligature_engraver	481
2.2.74	Laissez_vibrer_engraver	481
2.2.75	Ledger_line_engraver	481
2.2.76	Ligature_bracket_engraver	481
2.2.77	Lyric_engraver	481
2.2.78	Lyric_performer	482

2.2.79	Lyric_repeat_count_engraver.....	482
2.2.80	Mark_engraver.....	482
2.2.81	Mark_performer.....	483
2.2.82	Mark_tracking_translator.....	483
2.2.83	Measure_counter_engraver.....	484
2.2.84	Measure_grouping_engraver.....	484
2.2.85	Measure_spanner_engraver.....	484
2.2.86	Melody_engraver.....	485
2.2.87	Mensural_ligature_engraver.....	485
2.2.88	Merge_mmrest_numbers_engraver.....	485
2.2.89	Merge_rests_engraver.....	485
2.2.90	Metronome_mark_engraver.....	485
2.2.91	Midi_control_change_performer.....	486
2.2.92	Multi_measure_rest_engraver.....	487
2.2.93	New_fingering_engraver.....	487
2.2.94	Non_musical_script_column_engraver.....	488
2.2.95	Note_head_line_engraver.....	488
2.2.96	Note_heads_engraver.....	488
2.2.97	Note_name_engraver.....	488
2.2.98	Note_performer.....	489
2.2.99	Note_spacing_engraver.....	489
2.2.100	Optional_material_bracket_engraver.....	489
2.2.101	Ottava_spanner_engraver.....	489
2.2.102	Output_property_engraver.....	490
2.2.103	Page_turn_engraver.....	490
2.2.104	Paper_column_engraver.....	490
2.2.105	Parenthesis_engraver.....	491
2.2.106	Part_combine_engraver.....	491
2.2.107	Percent_repeat_engraver.....	492
2.2.108	Phrasing_slur_engraver.....	492
2.2.109	Piano_pedal_align_engraver.....	492
2.2.110	Piano_pedal_engraver.....	492
2.2.111	Piano_pedal_performer.....	493
2.2.112	Pitch_squash_engraver.....	493
2.2.113	Pitched_trill_engraver.....	494
2.2.114	Pure_from_neighbor_engraver.....	494
2.2.115	Repeat_acknowledge_engraver.....	494
2.2.116	Repeat_tie_engraver.....	494
2.2.117	Rest_collision_engraver.....	495
2.2.118	Rest_engraver.....	495
2.2.119	Rhythmic_column_engraver.....	495
2.2.120	Script_column_engraver.....	495
2.2.121	Script_engraver.....	496
2.2.122	Script_row_engraver.....	496
2.2.123	Separating_line_group_engraver.....	496
2.2.124	Show_control_points_engraver.....	496
2.2.125	Signum_repetitionis_engraver.....	497
2.2.126	Skip_typesetting_engraver.....	497
2.2.127	Slash_repeat_engraver.....	497
2.2.128	Slur_engraver.....	497
2.2.129	Slur_performer.....	498
2.2.130	Spacing_engraver.....	498

2.2.131	Span_arpeggio_engraver	498
2.2.132	Span_bar_engraver	498
2.2.133	Span_bar_stub_engraver	498
2.2.134	Span_stem_engraver	499
2.2.135	Spanner_break_forbid_engraver	499
2.2.136	Spanner_tracking_engraver	499
2.2.137	Staff_collecting_engraver	499
2.2.138	Staff_highlight_engraver	499
2.2.139	Staff_performer	500
2.2.140	Staff_symbol_engraver	500
2.2.141	Stanza_number_align_engraver	500
2.2.142	Stanza_number_engraver	500
2.2.143	Stem_engraver	500
2.2.144	System_start_delimiter_engraver	501
2.2.145	Tab_note_heads_engraver	501
2.2.146	Tab_staff_symbol_engraver	502
2.2.147	Tab_tie_follow_engraver	502
2.2.148	Tempo_performer	503
2.2.149	Text_engraver	503
2.2.150	Text_mark_engraver	503
2.2.151	Text_spanner_engraver	503
2.2.152	Tie_engraver	503
2.2.153	Tie_performer	504
2.2.154	Time_signature_engraver	504
2.2.155	Time_signature_performer	505
2.2.156	Timing_translator	505
2.2.157	Trill_spanner_engraver	506
2.2.158	Tuplet_engraver	506
2.2.159	Tweak_engraver	507
2.2.160	Vaticana_ligature_engraver	507
2.2.161	Vertical_align_engraver	507
2.2.162	Volta_engraver	507
2.3	Tunable context properties	508
2.4	Internal context properties	523

3 Backend 526

3.1	All layout objects	526
3.1.1	Accidental	526
3.1.2	AccidentalCautionary	527
3.1.3	AccidentalPlacement	528
3.1.4	AccidentalSuggestion	529
3.1.5	Ambitus	531
3.1.6	AmbitusAccidental	533
3.1.7	AmbitusLine	533
3.1.8	AmbitusNoteHead	534
3.1.9	Arpeggio	535
3.1.10	BalloonText	537
3.1.11	BarLine	538
3.1.12	BarNumber	542
3.1.13	BassFigure	544
3.1.14	BassFigureAlignment	544

3.1.15	BassFigureAlignmentPositioning.....	545
3.1.16	BassFigureBracket.....	546
3.1.17	BassFigureContinuation.....	547
3.1.18	BassFigureLine	548
3.1.19	Beam.....	549
3.1.20	BendAfter	551
3.1.21	BendSpanner.....	552
3.1.22	BreakAlignGroup.....	554
3.1.23	BreakAlignment	555
3.1.24	BreathingSign.....	557
3.1.25	CaesuraScript.....	559
3.1.26	CenteredBarNumber.....	561
3.1.27	CenteredBarNumberLineSpanner.....	562
3.1.28	ChordName	563
3.1.29	ChordSquare	564
3.1.30	Clef	565
3.1.31	ClefModifier.....	568
3.1.32	ClusterSpanner	569
3.1.33	ClusterSpannerBeacon	570
3.1.34	CodaMark	570
3.1.35	CombineTextScript.....	572
3.1.36	ControlPoint.....	575
3.1.37	ControlPolygon	576
3.1.38	CueClef	577
3.1.39	CueEndClef	580
3.1.40	Custos.....	583
3.1.41	Divisio.....	585
3.1.42	DotColumn	588
3.1.43	Dots	588
3.1.44	DoublePercentRepeat	590
3.1.45	DoublePercentRepeatCounter	591
3.1.46	DoubleRepeatSlash.....	592
3.1.47	DurationLine.....	593
3.1.48	DynamicLineSpanner.....	595
3.1.49	DynamicText.....	597
3.1.50	DynamicTextSpanner.....	599
3.1.51	Episema.....	600
3.1.52	FingerGlideSpanner.....	601
3.1.53	Fingering.....	603
3.1.54	FingeringColumn.....	605
3.1.55	Flag.....	605
3.1.56	Footnote	606
3.1.57	FretBoard	608
3.1.58	Glissando	610
3.1.59	GraceSpacing.....	611
3.1.60	GridChordName.....	612
3.1.61	GridLine	612
3.1.62	GridPoint	613
3.1.63	Hairpin.....	614
3.1.64	HorizontalBracket.....	615
3.1.65	HorizontalBracketText.....	617
3.1.66	InstrumentName	618

3.1.67	InstrumentSwitch.....	619
3.1.68	JumpScript.....	621
3.1.69	KeyCancellation.....	622
3.1.70	KeySignature.....	625
3.1.71	KievanLigature.....	628
3.1.72	LaissezVibrerTie.....	629
3.1.73	LaissezVibrerTieColumn.....	630
3.1.74	LedgerLineSpanner.....	631
3.1.75	LeftEdge.....	632
3.1.76	LigatureBracket.....	634
3.1.77	LyricExtender.....	635
3.1.78	LyricHyphen.....	636
3.1.79	LyricRepeatCount.....	637
3.1.80	LyricSpace.....	639
3.1.81	LyricText.....	640
3.1.82	MeasureCounter.....	641
3.1.83	MeasureGrouping.....	643
3.1.84	MeasureSpanner.....	644
3.1.85	MelodyItem.....	646
3.1.86	MensuralLigature.....	646
3.1.87	MetronomeMark.....	647
3.1.88	MultiMeasureRest.....	649
3.1.89	MultiMeasureRestNumber.....	650
3.1.90	MultiMeasureRestScript.....	652
3.1.91	MultiMeasureRestText.....	653
3.1.92	NonMusicalPaperColumn.....	655
3.1.93	NoteCollision.....	656
3.1.94	NoteColumn.....	657
3.1.95	NoteHead.....	658
3.1.96	NoteName.....	659
3.1.97	NoteSpacing.....	660
3.1.98	OptionalMaterialBracket.....	661
3.1.99	OttavaBracket.....	664
3.1.100	PaperColumn.....	665
3.1.101	Parentheses.....	666
3.1.102	PercentRepeat.....	667
3.1.103	PercentRepeatCounter.....	668
3.1.104	PhrasingSlur.....	670
3.1.105	PianoPedalBracket.....	672
3.1.106	RehearsalMark.....	673
3.1.107	RepeatSlash.....	675
3.1.108	RepeatTie.....	676
3.1.109	RepeatTieColumn.....	677
3.1.110	Rest.....	678
3.1.111	RestCollision.....	679
3.1.112	Script.....	679
3.1.113	ScriptColumn.....	680
3.1.114	ScriptRow.....	681
3.1.115	SectionLabel.....	681
3.1.116	SegnoMark.....	683
3.1.117	SignumRepetitionis.....	685
3.1.118	Slur.....	688

3.1.119	SostenutoPedal	690
3.1.120	SostenutoPedalLineSpanner	692
3.1.121	SpacingSpanner	693
3.1.122	SpanBar	694
3.1.123	SpanBarStub	695
3.1.124	StaffEllipsis	696
3.1.125	StaffGrouper	698
3.1.126	StaffHighlight	699
3.1.127	StaffSpacing	700
3.1.128	StaffSymbol	701
3.1.129	StanzaNumber	702
3.1.130	Stem	703
3.1.131	StemStub	705
3.1.132	StemTremolo	705
3.1.133	StringNumber	706
3.1.134	StrokeFinger	708
3.1.135	SustainPedal	710
3.1.136	SustainPedalLineSpanner	711
3.1.137	System	712
3.1.138	SystemStartBar	713
3.1.139	SystemStartBrace	714
3.1.140	SystemStartBracket	715
3.1.141	SystemStartSquare	716
3.1.142	TabNoteHead	717
3.1.143	TextMark	719
3.1.144	TextScript	721
3.1.145	TextSpanner	723
3.1.146	Tie	725
3.1.147	TieColumn	727
3.1.148	TimeSignature	727
3.1.149	TrillPitchAccidental	730
3.1.150	TrillPitchGroup	731
3.1.151	TrillPitchHead	733
3.1.152	TrillPitchParentheses	733
3.1.153	TrillSpanner	734
3.1.154	TupletBracket	736
3.1.155	TupletNumber	737
3.1.156	UnaCordaPedal	739
3.1.157	UnaCordaPedalLineSpanner	740
3.1.158	VaticanaLigature	741
3.1.159	VerticalAlignment	742
3.1.160	VerticalAxisGroup	742
3.1.161	VoiceFollower	744
3.1.162	VoltaBracket	745
3.1.163	VoltaBracketSpanner	747
3.1.164	VowelTransition	748
3.2	Graphical Object Interfaces	749
3.2.1	accidental-interface	749
3.2.2	accidental-participating-head-interface	750
3.2.3	accidental-placement-interface	750
3.2.4	accidental-suggestion-interface	751
3.2.5	accidental-switch-interface	751

3.2.6	align-interface	751
3.2.7	ambitus-interface	752
3.2.8	arpeggio-interface	752
3.2.9	axis-group-interface	753
3.2.10	balloon-interface	755
3.2.11	bar-line-interface	755
3.2.12	bar-number-interface	757
3.2.13	bass-figure-alignment-interface	757
3.2.14	bass-figure-interface	757
3.2.15	beam-interface	757
3.2.16	bend-after-interface	760
3.2.17	bend-interface	760
3.2.18	bezier-curve-interface	762
3.2.19	break-alignable-interface	762
3.2.20	break-aligned-interface	762
3.2.21	break-alignment-interface	764
3.2.22	breathing-sign-interface	765
3.2.23	caesura-script-interface	765
3.2.24	centered-bar-number-interface	765
3.2.25	centered-bar-number-line-spanner-interface	765
3.2.26	centered-spanner-interface	766
3.2.27	chord-name-interface	766
3.2.28	chord-square-interface	766
3.2.29	clef-interface	767
3.2.30	clef-modifier-interface	767
3.2.31	cluster-beacon-interface	767
3.2.32	cluster-interface	768
3.2.33	coda-mark-interface	768
3.2.34	control-point-interface	768
3.2.35	control-polygon-interface	768
3.2.36	custos-interface	769
3.2.37	dot-column-interface	769
3.2.38	dots-interface	770
3.2.39	duration-line-interface	770
3.2.40	dynamic-interface	771
3.2.41	dynamic-line-spanner-interface	771
3.2.42	dynamic-text-interface	771
3.2.43	dynamic-text-spanner-interface	771
3.2.44	enclosing-bracket-interface	771
3.2.45	episema-interface	772
3.2.46	figured-bass-continuation-interface	772
3.2.47	finger-glide-interface	772
3.2.48	finger-interface	773
3.2.49	fingering-column-interface	773
3.2.50	flag-interface	774
3.2.51	font-interface	774
3.2.52	footnote-interface	776
3.2.53	fret-diagram-interface	776
3.2.54	glissando-interface	778
3.2.55	grace-spacing-interface	778
3.2.56	gregorian-ligature-interface	778
3.2.57	grid-chord-name-interface	779

3.2.58	grid-line-interface	779
3.2.59	grid-point-interface	780
3.2.60	grob-interface	780
3.2.61	hairpin-interface	784
3.2.62	hara-kiri-group-spanner-interface	785
3.2.63	horizontal-bracket-interface	785
3.2.64	horizontal-bracket-text-interface	786
3.2.65	horizontal-line-spanner-interface	786
3.2.66	horizontal-script-interface	787
3.2.67	inline-accidental-interface	787
3.2.68	instrument-specific-markup-interface	788
3.2.69	item-interface	790
3.2.70	jump-script-interface	791
3.2.71	key-cancellation-interface	791
3.2.72	key-signature-interface	791
3.2.73	kievan-ligature-interface	792
3.2.74	ledger-line-spanner-interface	793
3.2.75	ledgered-interface	793
3.2.76	ligature-bracket-interface	793
3.2.77	ligature-head-interface	794
3.2.78	ligature-interface	794
3.2.79	line-interface	794
3.2.80	line-spanner-interface	795
3.2.81	lyric-extender-interface	796
3.2.82	lyric-hyphen-interface	797
3.2.83	lyric-interface	797
3.2.84	lyric-repeat-count-interface	797
3.2.85	lyric-space-interface	798
3.2.86	lyric-syllable-interface	798
3.2.87	mark-interface	798
3.2.88	measure-counter-interface	798
3.2.89	measure-grouping-interface	798
3.2.90	measure-spanner-interface	799
3.2.91	melody-spanner-interface	800
3.2.92	mensural-ligature-interface	800
3.2.93	metronome-mark-interface	801
3.2.94	multi-measure-interface	801
3.2.95	multi-measure-rest-interface	801
3.2.96	multi-measure-rest-number-interface	802
3.2.97	musical-paper-column-interface	802
3.2.98	non-musical-paper-column-interface	803
3.2.99	note-collision-interface	803
3.2.100	note-column-interface	804
3.2.101	note-head-interface	805
3.2.102	note-name-interface	806
3.2.103	note-spacing-interface	806
3.2.104	number-interface	806
3.2.105	optional-material-bracket-interface	807
3.2.106	ottava-bracket-interface	807
3.2.107	outside-staff-axis-group-interface	807
3.2.108	outside-staff-interface	808
3.2.109	paper-column-interface	809

3.2.110	parentheses-interface	809
3.2.111	passage-delimiter-interface	810
3.2.112	percent-repeat-interface	810
3.2.113	piano-pedal-bracket-interface	811
3.2.114	piano-pedal-interface	811
3.2.115	piano-pedal-script-interface	811
3.2.116	pitched-trill-interface	811
3.2.117	pure-from-neighbor-interface	812
3.2.118	rehearsal-mark-interface	812
3.2.119	rest-collision-interface	812
3.2.120	rest-interface	812
3.2.121	rhythmic-grob-interface	813
3.2.122	rhythmic-head-interface	813
3.2.123	script-column-interface	813
3.2.124	script-interface	814
3.2.125	section-label-interface	815
3.2.126	segno-mark-interface	815
3.2.127	self-alignment-interface	815
3.2.128	semi-tie-column-interface	816
3.2.129	semi-tie-interface	817
3.2.130	separation-item-interface	818
3.2.131	side-position-interface	818
3.2.132	signum-repetitionis-interface	819
3.2.133	slur-interface	820
3.2.134	spaceable-grob-interface	823
3.2.135	spacing-interface	824
3.2.136	spacing-options-interface	824
3.2.137	spacing-spanner-interface	824
3.2.138	span-bar-interface	825
3.2.139	span-bar-stub-interface	826
3.2.140	spanner-interface	826
3.2.141	staff-ellipsis-interface	827
3.2.142	staff-grouper-interface	827
3.2.143	staff-highlight-interface	828
3.2.144	staff-spacing-interface	828
3.2.145	staff-symbol-interface	828
3.2.146	staff-symbol-referencer-interface	829
3.2.147	stanza-number-interface	830
3.2.148	stem-interface	830
3.2.149	stem-tremolo-interface	832
3.2.150	sticky-grob-interface	833
3.2.151	string-number-interface	833
3.2.152	stroke-finger-interface	833
3.2.153	system-interface	833
3.2.154	system-start-delimiter-interface	834
3.2.155	system-start-text-interface	835
3.2.156	tab-note-head-interface	835
3.2.157	text-interface	837
3.2.158	text-mark-interface	837
3.2.159	text-script-interface	837
3.2.160	tie-column-interface	838
3.2.161	tie-interface	838

3.2.162	time-signature-interface.....	841
3.2.163	trill-pitch-accidental-interface.....	842
3.2.164	trill-spanner-interface.....	842
3.2.165	tuplet-bracket-interface.....	843
3.2.166	tuplet-number-interface.....	845
3.2.167	unbreakable-spanner-interface.....	845
3.2.168	vaticana-ligature-interface.....	845
3.2.169	volta-bracket-interface.....	846
3.2.170	volta-interface.....	847
3.3	User backend properties.....	847
3.4	Internal backend properties.....	872
4	Scheme functions.....	880
Appendix A	Indices.....	944
A.1	Concept index.....	944
A.2	Function index.....	944

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 49), dynamic-event (page 52), music-event (page 54), and StreamEvent (page 58).

Accepted by: Dynamic_engraver (page 470), and Dynamic_performer (page 470).

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 49), music-event (page 54), and StreamEvent (page 58).

Accepted by: Bar_engraver (page 453), and Jump_engraver (page 478).

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 49), mark-event (page 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: `Mark_tracking_translator` (page 483).

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 49), `music-event` (page 54), `StreamEvent` (page 58), and `structural-event` (page 59).

Accepted by: `Timing_translator` (page 505).

Properties:

name (symbol):

`'AlternativeEvent'`

Name of this music object.

types (list):

`'(alternative-event structural-event 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 49), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Balloon_engraver` (page 453).

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 49), `layout-instruction-event` (page 53), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Output_property_engraver` (page 490).

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 54), and `StreamEvent` (page 58).

Accepted by: `Arpeggio_engraver` (page 452).

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 50), `music-event` (page 54), `script-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Beat_engraver` (page 458), `Beat_performer` (page 458), `Drum_note_performer` (page 468), `Note_performer` (page 489), and `Script_engraver` (page 496).

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 BarCheckEvent

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

Event classes: `bar-check-event` (page 50), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Timing_translator` (page 505).

Properties:

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

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

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 50), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Timing_translator` (page 505).

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 50), `music-event` (page 54), `rhythmic-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Figured_bass_engraver` (page 471).

Properties:

name (symbol):

`'BassFigureEvent`

Name of this music object.

types (list):

`'(event rhythmic-event bass-figure-event)`

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

1.1.13 BeamEvent

Start or stop a beam.

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

Event classes: `beam-event` (page 50), `music-event` (page 54), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Beam_engraver` (page 457), `Beam_performer` (page 458), and `Grace_beam_engraver` (page 475).

Properties:

name (symbol):

`'BeamEvent`

Name of this music object.

types (list):

`'(post-event event beam-event span-event)`

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

1.1.14 BeamForbidEvent

Specify that a note may not auto-beamed.

Event classes: `beam-forbid-event` (page 50), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Auto_beam_engraver` (page 452), and `Grace_auto_beam_engraver` (page 474).

Properties:

name (symbol):

`'BeamForbidEvent`

Name of this music object.

types (list):

`'(post-event event beam-forbid-event)`

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

1.1.15 BendAfterEvent

A drop/fall/doit jazz articulation.

Event classes: bend-after-event (page 50), music-event (page 54), and StreamEvent (page 58).

Accepted by: Bend_engraver (page 459).

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 50), music-event (page 54), span-event (page 57), and StreamEvent (page 58).

Accepted by: Bend_spanner_engraver (page 459).

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 50), break-span-event (page 51), music-event (page 54), and StreamEvent (page 58).

Accepted by: Dynamic_engraver (page 470).

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 51), *music-event* (page 54), and *StreamEvent* (page 58).

Accepted by: *Breathing_sign_engraver* (page 460), and *Note_performer* (page 489).

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 51), *music-event* (page 54), and *StreamEvent* (page 58).

Accepted by: *Bar_engraver* (page 453), *Caesura_engraver* (page 460), and *Divisio_engraver* (page 467).

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 51), *melodic-event* (page 54), *music-event* (page 54), *rhythmic-event* (page 56), and *StreamEvent* (page 58).

Accepted by: *Cluster_spanner_engraver* (page 463).

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 51), music-event (page 54), StreamEvent (page 58), and structural-event (page 59).

Accepted by: Bar_engraver (page 453), and Mark_tracking_translator (page 483).

Properties:

```

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

types (list):
  '(coda-mark-event structural-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 51), music-event (page 54), and StreamEvent (page 58).

Accepted by: Extender_engraver (page 471).

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 51), music-event (page 54), span-dynamic-event (page 57), span-event (page 57), and StreamEvent (page 58).

Accepted by: Dynamic_engraver (page 470), and Dynamic_performer (page 470).

Properties:

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

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

```
crescendo-event
event)
```

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

1.1.26 DalSegnoEvent

Add a *D.S.* or similar instruction.

Event classes: `dal-segno-event` (page 51), `music-event` (page 54), `StreamEvent` (page 58), and `structural-event` (page 59).

Accepted by: `Bar_engraver` (page 453), `Jump_engraver` (page 478), and `Volta_engraver` (page 507).

Properties:

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

types (list):
  '(dal-segno-event structural-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 51), `music-event` (page 54), `span-dynamic-event` (page 57), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Dynamic_engraver` (page 470), and `Dynamic_performer` (page 470).

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 51), `music-event` (page 54), `rhythmic-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Double_percent_repeat_engraver` (page 468).

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 52), music-event (page 54), and StreamEvent (page 58).

Accepted by: Duration_line_engraver (page 469).

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 52), music-event (page 54), span-event (page 57), and StreamEvent (page 58).

Accepted by: Episema_engraver (page 471).

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)

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 52), music-event (page 54), and StreamEvent (page 58).

Accepted by: Extender_engraver (page 471).

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 52), `music-event` (page 54), `StreamEvent` (page 58), and `structural-event` (page 59).

Accepted by: `Bar_engraver` (page 453), `Divisio_engraver` (page 467), `Jump_engraver` (page 478), `Timing_translator` (page 505), and `Volta_engraver` (page 507).

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 52), `music-event` (page 54), `span-event` (page 57), and `StreamEvent` (page 58).

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 52), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Fingering_engraver` (page 472), `Fretboard_engraver` (page 473), and `Tab_note_heads_engraver` (page 501).

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 52), music-event (page 54), and StreamEvent (page 58).

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 52), music-event (page 54), and StreamEvent (page 58).

Accepted by: Glissando_engraver (page 474).

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 52), music-event (page 54), and StreamEvent (page 58).

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 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: Hyphen_engraver (page 477).

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 InitialContextMusic

Enter the initial context of the argument and ignore the rest of it.

Syntax: \initialContextFrom *music*

Properties:

iterator-ctor (procedure):
 ly:initial-context-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):

'InitialContextMusic

Name of this music object.

to-relative-callback (procedure):

ly:relative-octave-music::no-relative-callback

How to transform a piece of music to relative pitches.

types (list):

'(initial-context-music)

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

1.1.43 KeyChangeEvent

Change the key signature.

Syntax: \key *name scale*

Event classes: key-change-event (page 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: Key_engraver (page 479), and Key_performer (page 480).

Properties:

name (symbol):

'KeyChangeEvent

Name of this music object.

to-relative-callback (procedure):

#<procedure at lily/define-music-types.scm:337:33 (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.

1.1.44 LabelEvent

Place a bookmarking label.

Event classes: label-event (page 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: Paper_column_engraver (page 490).

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.

1.1.45 LaissezVibrerEvent

Don't damp this chord.

Syntax: *note*\laissezVibrer

Event classes: laissez-vibrer-event (page 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: Laissez_vibrer_engraver (page 481).

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.

1.1.46 LigatureEvent

Start or end a ligature.

Event classes: ligature-event (page 53), music-event (page 54), span-event (page 57), and StreamEvent (page 58).

Accepted by: Kievan_ligature_engraver (page 481), Ligature_bracket_engraver (page 481), Mensural_ligature_engraver (page 485), and Vaticana_ligature_engraver (page 507).

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.

1.1.47 LineBreakEvent

Allow, forbid or force a line break.

Event classes: break-event (page 50), line-break-event (page 53), music-event (page 54), and StreamEvent (page 58).

Accepted by: Page_turn_engraver (page 490), and Paper_column_engraver (page 490).

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.

1.1.48 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.

1.1.49 LyricEvent

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

Event classes: lyric-event (page 53), music-event (page 54), rhythmic-event (page 56), and StreamEvent (page 58).

Accepted by: Lyric_engraver (page 481), and Lyric_performer (page 482).

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.

1.1.50 MeasureCounterEvent

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

Event classes: `measure-counter-event` (page 53), `music-event` (page 54), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Measure_counter_engraver` (page 484).

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.
```

1.1.51 MeasureSpannerEvent

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

Event classes: `measure-spanner-event` (page 54), `music-event` (page 54), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Measure_spanner_engraver` (page 484).

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.
```

1.1.52 MultiMeasureArticulationEvent

Articulations on multi-measure rests.

Event classes: `multi-measure-articulation-event` (page 54), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Multi_measure_rest_engraver` (page 487).

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.
```

1.1.53 MultiMeasureRestEvent

Used internally by MultiMeasureRestMusic to signal rests.

Event classes: `general-rest-event` (page 52), `multi-measure-rest-event` (page 54), `music-event` (page 54), `rhythmic-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Current_chord_text_engraver` (page 466), and `Multi_measure_rest_engraver` (page 487).

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.

1.1.54 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.

1.1.55 MultiMeasureTextEvent

Texts on multi-measure rests.

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

Event classes: `multi-measure-text-event` (page 54), `music-event` (page 54), and `StreamEvent` (page 58).

Accepted by: `Multi_measure_rest_engraver` (page 487).

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.

1.1.56 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.

1.1.57 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 54), music-event (page 54), note-event (page 55), rhythmic-event (page 56), and StreamEvent (page 58).

Accepted by: Beat_engraver (page 458), Beat_performer (page 458), Bend_spanner_engraver (page 459), Completion_heads_engraver (page 463), Current_chord_text_engraver (page 466), Drum_note_performer (page 468), Drum_notes_engraver (page 468), Finger_glide_engraver (page 472), Fretboard_engraver (page 473), Note_heads_engraver (page 488), Note_name_engraver (page 488), Note_performer (page 489), Part_combine_engraver (page 491), Phrasing_slur_engraver (page 492), Slur_engraver (page 497), and Tab_note_heads_engraver (page 501).

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.

1.1.58 NoteGroupingEvent

Start or stop grouping brackets.

Event classes: `music-event` (page 54), `note-grouping-event` (page 55), and `StreamEvent` (page 58).

Accepted by: `Horizontal_bracket_engraver` (page 477).

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.

1.1.59 OptionalMaterialEvent

Start or stop optional material.

Event classes: `music-event` (page 54), `optional-material-event` (page 55), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Optional_material_bracket_engraver` (page 489).

Properties:

`name (symbol):`

`'OptionalMaterialEvent`

Name of this music object.

`types (list):`

`'(optional-material-event span-event event)`

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

1.1.60 OttavaEvent

Start or stop an ottava bracket.

Event classes: `music-event` (page 54), `ottava-event` (page 55), and `StreamEvent` (page 58).

Accepted by: `Ottava_spanner_engraver` (page 489).

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.

1.1.61 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.

1.1.62 PageBreakEvent

Allow, forbid or force a page break.

Event classes: `break-event` (page 50), `music-event` (page 54), `page-break-event` (page 55), and `StreamEvent` (page 58).

Accepted by: `Page_turn_engraver` (page 490), and `Paper_column_engraver` (page 490).

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.

1.1.63 PageTurnEvent

Allow, forbid or force a page turn.

Event classes: `break-event` (page 50), `music-event` (page 54), `page-turn-event` (page 55), and `StreamEvent` (page 58).

Accepted by: `Page_turn_engraver` (page 490), and `Paper_column_engraver` (page 490).

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.

1.1.64 PartCombineEvent

Announce partcombine status change, such as aDue, soloI, or soloII, based on value of property part-combine-status.

Event classes: music-event (page 54), part-combine-event (page 55), and StreamEvent (page 58).

Accepted by: Part_combine_engraver (page 491).

Properties:

name (symbol):

'PartCombineEvent

Name of this music object.

types (list):

'(event part-combine-event)

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

1.1.65 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.

1.1.66 PartialEvent

An event announcing a partial measure.

Event classes: music-event (page 54), partial-event (page 55), and StreamEvent (page 58).

Accepted by: Timing_translator (page 505).

Properties:

name (symbol):

'PartialEvent

Name of this music object.

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

1.1.67 PartialSet

Create an anacrusis or upbeat (partial measure).

Properties:

elements-callback (procedure):
 make-partial-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: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.

1.1.68 PercentEvent

Used internally to signal percent repeats.

Event classes: music-event (page 54), percent-event (page 56), and StreamEvent (page 58).

Accepted by: Percent_repeat_engraver (page 492).

Properties:

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

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

1.1.69 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.

1.1.70 **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 54), `pes-or-flexa-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Vaticana_ligature_engraver` (page 507).

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.

1.1.71 **PhrasingSlurEvent**

Start or end phrasing slur.

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

Event classes: `music-event` (page 54), `phrasing-slur-event` (page 56), `span-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Phrasing_slur_engraver` (page 492).

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.

1.1.72 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.

1.1.73 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.

1.1.74 PropertyUnset

Restore the default setting for a context property. See Section 1.1.73 [PropertySet], page 27.

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.

1.1.75 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.

1.1.76 RehearsalMarkEvent

Insert a rehearsal mark.

Syntax: `\mark marker`

Example: `\mark 3`

Event classes: mark-event (page 53), music-event (page 54), rehearsal-mark-event (page 56), and StreamEvent (page 58).

Accepted by: Mark_tracking_translator (page 483).

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.

1.1.77 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.

1.1.78 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.

1.1.79 RepeatSlashEvent

Used internally to signal beat repeats.

Event classes: music-event (page 54), repeat-slash-event (page 56), rhythmic-event (page 56), and StreamEvent (page 58).

Accepted by: Slash_repeat_engraver (page 497).

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.
```

1.1.80 RepeatTieEvent

Ties for starting a second volta bracket.

Event classes: `music-event` (page 54), `repeat-tie-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Repeat_tie_engraver` (page 494).

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.
```

1.1.81 RestEvent

A Rest.

Syntax: `r4` for a quarter rest.

Event classes: `general-rest-event` (page 52), `music-event` (page 54), `rest-event` (page 56), `rhythmic-event` (page 56), and `StreamEvent` (page 58).

Accepted by: `Completion_rest_engraver` (page 464), `Current_chord_text_engraver` (page 466), `Figured_bass_engraver` (page 471), and `Rest_engraver` (page 495).

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.
```

1.1.82 RevertProperty

The opposite of Section 1.1.61 [OverrideProperty], page 23: 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.
```

1.1.83 ScriptEvent

Add an articulation mark to a note.

Event classes: music-event (page 54), script-event (page 56), and StreamEvent (page 58).

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.
```

1.1.84 SectionEvent

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

Event classes: music-event (page 54), section-event (page 57), StreamEvent (page 58), and structural-event (page 59).

Accepted by: Bar_engraver (page 453), and Divisio_engraver (page 467).

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.
```


1.1.85 SectionLabelEvent

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

Event classes: `music-event` (page 54), `section-label-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Mark_tracking_translator` (page 483).

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.

1.1.86 SegnoMarkEvent

Add a segno mark or bar line.

Event classes: `music-event` (page 54), `segno-mark-event` (page 57), `StreamEvent` (page 58), and `structural-event` (page 59).

Accepted by: `Bar_engraver` (page 453), and `Mark_tracking_translator` (page 483).

Properties:

name (symbol):

`'SegnoMarkEvent`

Name of this music object.

types (list):

`'(segno-mark-event structural-event event)`

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

1.1.87 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.

1.1.88 SequentialAlternativeMusic

Repeat alternatives in sequence.

Syntax: \alternative { *alternatives* }

Properties:

elements-callback (procedure):

#<procedure at lily/define-music-types.scm:647:30 (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.

1.1.89 SequentialMusic

Music expressions concatenated.

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

Properties:

elements-callback (procedure):

#<procedure at lily/define-music-types.scm:660:30 (m)>

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

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

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

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

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

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

1.1.90 SimultaneousMusic

Music playing together.

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

Properties:

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

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

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

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

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

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

1.1.91 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 54), `rhythmic-event` (page 56), `skip-event` (page 57), and `StreamEvent` (page 58).

Not accepted by any engraver or performer.

Properties:

`iterator-ctor` (procedure):

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

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

`name` (symbol):

`'SkipEvent`

Name of this music object.

`types` (list):

`'(event rhythmic-event skip-event)`

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

1.1.92 SkipMusic

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

Syntax: `\skip duration`

Properties:

`iterator-ctor` (procedure):

`ly:simple-music-iterator::constructor`

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

`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.

1.1.93 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.

1.1.94 SlurEvent

Start or end slur.

Syntax: *note*(and *note*)

Event classes: music-event (page 54), slur-event (page 57), span-event (page 57), and StreamEvent (page 58).

Accepted by: Slur_engraver (page 497), and Slur_performer (page 498).

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.

1.1.95 SostenutoEvent

Depress or release sostenuto pedal.

Event classes: music-event (page 54), pedal-event (page 56), sostenuto-event (page 57), span-event (page 57), and StreamEvent (page 58).

Accepted by: Piano_pedal_engraver (page 492), and Piano_pedal_performer (page 493).

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.

1.1.96 SpacingSectionEvent

Start a new spacing section.

Event classes: `music-event` (page 54), `spacing-section-event` (page 57), and `StreamEvent` (page 58).

Accepted by: `Spacing_engraver` (page 498).

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.

1.1.97 SpanEvent

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

Event classes: `music-event` (page 54), `span-event` (page 57), and `StreamEvent` (page 58).

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.

1.1.98 StaffHighlightEvent

Start or stop a staff highlight.

Syntax: `\staffHighlight`, `\stopStaffHighlight`.

Event classes: `music-event` (page 54), `span-event` (page 57), `staff-highlight-event` (page 58), and `StreamEvent` (page 58).

Accepted by: `Staff_highlight_engraver` (page 499).

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.

1.1.99 StaffSpanEvent

Start or stop a staff symbol.

Event classes: `music-event` (page 54), `span-event` (page 57), `staff-span-event` (page 58), and `StreamEvent` (page 58).

Accepted by: `Staff_symbol_engraver` (page 500).

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.

1.1.100 StringNumberEvent

Specify on which string to play this note.

Syntax: `\number`

Event classes: `music-event` (page 54), `StreamEvent` (page 58), and `string-number-event` (page 58).

Accepted by: `Bend_spanner_engraver` (page 459), `Fretboard_engraver` (page 473), and `Tab_note_heads_engraver` (page 501).

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.

1.1.101 StrokeFingerEvent

Specify with which finger to pluck a string.

Syntax: `\rightHandFinger text`

Event classes: `music-event` (page 54), `StreamEvent` (page 58), and `stroke-finger-event` (page 58).

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.

1.1.102 SustainEvent

Depress or release sustain pedal.

Event classes: music-event (page 54), pedal-event (page 56), span-event (page 57), StreamEvent (page 58), and sustain-event (page 59).

Accepted by: Piano_pedal_engraver (page 492), and Piano_pedal_performer (page 493).

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.

1.1.103 TempoChangeEvent

A metronome mark or tempo indication.

Event classes: music-event (page 54), StreamEvent (page 58), and tempo-change-event (page 59).

Accepted by: Metronome_mark_engraver (page 485).

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.

1.1.104 TextMarkEvent

A textual mark.

Syntax: \textMark *markup* or \textEndMark *markup*.

Event classes: music-event (page 54), StreamEvent (page 58), and text-mark-event (page 59).

Accepted by: Text_mark_engraver (page 503).

Properties:

name (symbol):

'TextMarkEvent

Name of this music object.

types (list):

'(text-mark-event event)

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

1.1.105 TextScriptEvent

Print text.

Event classes: music-event (page 54), script-event (page 56), StreamEvent (page 58), and text-script-event (page 59).

Accepted by: Text_engraver (page 503).

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.

1.1.106 TextSpanEvent

Start a text spanner, for example, an octavation.

Event classes: music-event (page 54), span-event (page 57), StreamEvent (page 58), and text-span-event (page 59).

Accepted by: Text_spanner_engraver (page 503).

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.

1.1.107 TieEvent

A tie.

Syntax: *note*--

Event classes: music-event (page 54), StreamEvent (page 58), and tie-event (page 59).

Accepted by: Drum_note_performer (page 468), Note_performer (page 489), Tie_engraver (page 503), and Tie_performer (page 504).

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.

1.1.108 TimeScaledMusic

Multiply durations, as in tuplets.

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

Properties:

```

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

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

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

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

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

1.1.109 TimeSignatureEvent

An event created when setting a new time signature

Event classes: `music-event` (page 54), `StreamEvent` (page 58), and `time-signature-event` (page 59).

Accepted by: `Time_signature_engraver` (page 504), and `Time_signature_performer` (page 505).

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.
```

1.1.110 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 param-
  eter.
```

```

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.

```

1.1.111 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.

```

1.1.112 TremoloEvent

Unmeasured tremolo.

Event classes: `music-event` (page 54), `StreamEvent` (page 58), and `tremolo-event` (page 59).

Accepted by: `Stem_engraver` (page 500).

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.

1.1.113 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.

1.1.114 TremoloSpanEvent

Tremolo over two stems.

Event classes: music-event (page 54), span-event (page 57), StreamEvent (page 58), and tremolo-span-event (page 59).

Accepted by: Chord_tremolo_engraver (page 462).

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.

1.1.115 TrillSpanEvent

Start a trill spanner.

Event classes: `music-event` (page 54), `span-event` (page 57), `StreamEvent` (page 58), and `trill-span-event` (page 60).

Accepted by: `Trill_spanner_engraver` (page 506).

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.

1.1.116 TupletSpanEvent

Used internally to signal where tuplet brackets start and stop.

Event classes: `music-event` (page 54), `span-event` (page 57), `StreamEvent` (page 58), and `tuplet-span-event` (page 60).

Accepted by: `Tuplet_engraver` (page 506).

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.

1.1.117 UnaCordaEvent

Depress or release una-corda pedal.

Event classes: `music-event` (page 54), `pedal-event` (page 56), `span-event` (page 57), `StreamEvent` (page 58), and `una-corda-event` (page 60).

Accepted by: `Piano_pedal_engraver` (page 492), and `Piano_pedal_performer` (page 493).

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.

1.1.118 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.
```

1.1.119 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.
```

1.1.120 UnrelativableMusic

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

Properties:

```

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

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

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

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

to-relative-callback (procedure):
  ly:relative-octave-music::no-relative-callback
  How to transform a piece of music to relative pitches.

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

```

1.1.121 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.

```

1.1.122 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 54), `StreamEvent` (page 58), `structural-event` (page 59), and `volta-repeat-end-event` (page 60).

Accepted by: `Divisio_engraver` (page 467), `Lyric_repeat_count_engraver` (page 482), `Repeat_acknowledge_engraver` (page 494), and `Signum_repetitionis_engraver` (page 497).

Properties:

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

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

1.1.123 VoltaRepeatStartEvent

Signal the start of a volta-style repeat.

Event classes: music-event (page 54), StreamEvent (page 58), structural-event (page 59), and volta-repeat-start-event (page 60).

Accepted by: Divisio_engraver (page 467), and Repeat_acknowledge_engraver (page 494).

Properties:

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

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

1.1.124 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.

1.1.125 VoltaSpanEvent

Used internally to signal where volta brackets start and stop.

Event classes: music-event (page 54), span-event (page 57), StreamEvent (page 58), and volta-span-event (page 60).

Accepted by: Volta_engraver (page 507).

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.

1.1.126 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.

types (list):

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

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

1.1.127 VowelTransitionEvent

A vowel transition between lyric syllables.

Event classes: `music-event` (page 54), `StreamEvent` (page 58), and `vowel-transition-event` (page 60).

Accepted by: `Hyphen_engraver` (page 477).

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 470), and `Dynamic_performer` (page 470).

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 453), and `Jump_engraver` (page 478).

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 483).

1.2.4 alternative-event

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

Accepted by: `Timing_translator` (page 505).

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 453).

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 490).

1.2.7 arpeggio-event

Music event type `arpeggio-event` is in music objects of type `ArpeggioEvent` (page 3).

Accepted by: `Arpeggio_engraver` (page 452).

1.2.8 articulation-event

Music event type articulation-event is in music objects of type ArticulationEvent (page 3).

Accepted by: Beat_engraver (page 458), Beat_performer (page 458), Drum_note_performer (page 468), Note_performer (page 489), and Script_engraver (page 496).

1.2.9 bar-check-event

Music event type bar-check-event is in music objects of type BarCheckEvent (page 4).

Accepted by: Timing_translator (page 505).

1.2.10 bar-event

Music event type bar-event is in music objects of type BarEvent (page 4).

Accepted by: Timing_translator (page 505).

1.2.11 bass-figure-event

Music event type bass-figure-event is in music objects of type BassFigureEvent (page 5).

Accepted by: Figured_bass_engraver (page 471).

1.2.12 beam-event

Music event type beam-event is in music objects of type BeamEvent (page 5).

Accepted by: Beam_engraver (page 457), Beam_performer (page 458), and Grace_beam_engraver (page 475).

1.2.13 beam-forbid-event

Music event type beam-forbid-event is in music objects of type BeamForbidEvent (page 5).

Accepted by: Auto_beam_engraver (page 452), and Grace_auto_beam_engraver (page 474).

1.2.14 bend-after-event

Music event type bend-after-event is in music objects of type BendAfterEvent (page 6).

Accepted by: Bend_engraver (page 459).

1.2.15 bend-span-event

Music event type bend-span-event is in music objects of type BendSpanEvent (page 6).

Accepted by: Bend_spanner_engraver (page 459).

1.2.16 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 470).

1.2.17 break-event

Music event type break-event is in music objects of type LineBreakEvent (page 17), PageBreakEvent (page 23), and PageTurnEvent (page 23).

Accepted by: Page_turn_engraver (page 490), and Paper_column_engraver (page 490).

1.2.18 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.

1.2.19 breathing-event

Music event type breathing-event is in music objects of type BreathingEvent (page 7).

Accepted by: Breathing_sign_engraver (page 460), and Note_performer (page 489).

1.2.20 caesura-event

Music event type caesura-event is in music objects of type CaesuraEvent (page 7).

Accepted by: Bar_engraver (page 453), Caesura_engraver (page 460), and Divisio_engraver (page 467).

1.2.21 cluster-note-event

Music event type cluster-note-event is in music objects of type ClusterNoteEvent (page 7).

Accepted by: Cluster_spanner_engraver (page 463).

1.2.22 coda-mark-event

Music event type coda-mark-event is in music objects of type CodaMarkEvent (page 8).

Accepted by: Bar_engraver (page 453), and Mark_tracking_translator (page 483).

1.2.23 completize-extender-event

Music event type completize-extender-event is in music objects of type CompletizeExtenderEvent (page 8).

Accepted by: Extender_engraver (page 471).

1.2.24 crescendo-event

Music event type crescendo-event is in music objects of type CrescendoEvent (page 9).

Accepted by: Dynamic_performer (page 470).

1.2.25 dal-segno-event

Music event type dal-segno-event is in music objects of type DalSegnoEvent (page 10).

Accepted by: Bar_engraver (page 453), Jump_engraver (page 478), and Volta_engraver (page 507).

1.2.26 decrescendo-event

Music event type decrescendo-event is in music objects of type DecrescendoEvent (page 10).

Accepted by: Dynamic_performer (page 470).

1.2.27 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 468).

1.2.28 duration-line-event

Music event type duration-line-event is in music objects of type DurationLineEvent (page 11).

Accepted by: Duration_line_engraver (page 469).

1.2.29 dynamic-event

Music event type dynamic-event is in music objects of type AbsoluteDynamicEvent (page 1).

Not accepted by any engraver or performer.

1.2.30 episema-event

Music event type episema-event is in music objects of type EpisemaEvent (page 11).

Accepted by: Episema_engraver (page 471).

1.2.31 extender-event

Music event type extender-event is in music objects of type ExtenderEvent (page 12).

Accepted by: Extender_engraver (page 471).

1.2.32 fine-event

Music event type fine-event is in music objects of type FineEvent (page 13).

Accepted by: Bar_engraver (page 453), Divisio_engraver (page 467), Jump_engraver (page 478), Timing_translator (page 505), and Volta_engraver (page 507).

1.2.33 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.

1.2.34 fingering-event

Music event type fingering-event is in music objects of type FingeringEvent (page 13).

Accepted by: Fingering_engraver (page 472), Fretboard_engraver (page 473), and Tab_note_heads_engraver (page 501).

1.2.35 footnote-event

Music event type footnote-event is in music objects of type FootnoteEvent (page 14).

Not accepted by any engraver or performer.

1.2.36 general-rest-event

Music event type general-rest-event is in music objects of type MultiMeasureRestEvent (page 20), and RestEvent (page 30).

Accepted by: Current_chord_text_engraver (page 466).

1.2.37 glissando-event

Music event type glissando-event is in music objects of type GlissandoEvent (page 14).

Accepted by: Glissando_engraver (page 474).

1.2.38 harmonic-event

Music event type harmonic-event is in music objects of type HarmonicEvent (page 15).

Not accepted by any engraver or performer.

1.2.39 hyphen-event

Music event type hyphen-event is in music objects of type HyphenEvent (page 15).

Accepted by: Hyphen_engraver (page 477).

1.2.40 key-change-event

Music event type key-change-event is in music objects of type KeyChangeEvent (page 16).

Accepted by: Key_engraver (page 479), and Key_performer (page 480).

1.2.41 label-event

Music event type label-event is in music objects of type LabelEvent (page 16).

Accepted by: Paper_column_engraver (page 490).

1.2.42 laissez-vibrer-event

Music event type laissez-vibrer-event is in music objects of type LaissezVibrerEvent (page 17).

Accepted by: Laissez_vibrer_engraver (page 481).

1.2.43 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.

1.2.44 ligature-event

Music event type ligature-event is in music objects of type LigatureEvent (page 17).

Accepted by: Kievan_ligature_engraver (page 481), Ligature_bracket_engraver (page 481), Mensural_ligature_engraver (page 485), and Vaticana_ligature_engraver (page 507).

1.2.45 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.

1.2.46 lyric-event

Music event type lyric-event is in music objects of type LyricEvent (page 18).

Accepted by: Lyric_engraver (page 481), and Lyric_performer (page 482).

1.2.47 mark-event

Music event type mark-event is in music objects of type AdHocMarkEvent (page 1), and RehearsalMarkEvent (page 28).

Not accepted by any engraver or performer.

1.2.48 measure-counter-event

Music event type measure-counter-event is in music objects of type MeasureCounterEvent (page 19).

Accepted by: Measure_counter_engraver (page 484).

1.2.49 measure-spanner-event

Music event type `measure-spanner-event` is in music objects of type `MeasureSpannerEvent` (page 19).

Accepted by: `Measure_spanner_engraver` (page 484).

1.2.50 melodic-event

Music event type `melodic-event` is in music objects of type `ClusterNoteEvent` (page 7), and `NoteEvent` (page 21).

Not accepted by any engraver or performer.

1.2.51 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 487).

1.2.52 multi-measure-rest-event

Music event type `multi-measure-rest-event` is in music objects of type `MultiMeasureRestEvent` (page 20).

Accepted by: `Multi_measure_rest_engraver` (page 487).

1.2.53 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 487).

1.2.54 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), `BarCheckEvent` (page 4), `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 16), `LabelEvent` (page 16), `LaissezVibrerEvent` (page 17), `LigatureEvent` (page 17), `LineBreakEvent` (page 17), `LyricEvent` (page 18), `MeasureCounterEvent` (page 19), `MeasureSpannerEvent` (page 19), `MultiMeasureArticulationEvent` (page 19), `MultiMeasureRestEvent` (page 20), `MultiMeasureTextEvent` (page 20), `NoteEvent` (page 21), `NoteGroupingEvent` (page 22), `OptionalMaterialEvent` (page 22), `OttavaEvent` (page 22), `PageBreakEvent` (page 23), `PageTurnEvent` (page 23), `PartCombineEvent` (page 24), `PartialEvent` (page 24), `PercentEvent` (page 25), `PesOrFlexaEvent` (page 26), `PhrasingSlurEvent` (page 26), `RehearsalMarkEvent` (page 28), `RepeatSlashEvent` (page 29), `RepeatTieEvent` (page 30), `RestEvent` (page 30), `ScriptEvent` (page 31), `SectionEvent` (page 31), `SectionLabelEvent` (page 32), `SegnoMarkEvent` (page 32), `SkipEvent` (page 35), `SlurEvent` (page 36), `SostenutoEvent` (page 36), `SpacingSectionEvent` (page 37), `SpanEvent` (page 37), `StaffHighlightEvent` (page 37), `StaffSpanEvent` (page 38), `StringNumberEvent` (page 38),

StrokeFingerEvent (page 38), SustainEvent (page 39), TempoChangeEvent (page 39), TextMarkEvent (page 39), TextScriptEvent (page 40), TextSpanEvent (page 40), TieEvent (page 40), TimeSignatureEvent (page 41), TremoloEvent (page 42), TremoloSpanEvent (page 43), TrillSpanEvent (page 44), TupletSpanEvent (page 44), UnaCordaEvent (page 44), VoltaRepeatEndEvent (page 46), VoltaRepeatStartEvent (page 47), VoltaSpanEvent (page 48), and VowelTransitionEvent (page 49).

Not accepted by any engraver or performer.

1.2.55 note-event

Music event type note-event is in music objects of type NoteEvent (page 21).

Accepted by: Beat_engraver (page 458), Beat_performer (page 458), Bend_spanner_engraver (page 459), Completion_heads_engraver (page 463), Current_chord_text_engraver (page 466), Drum_note_performer (page 468), Drum_notes_engraver (page 468), Finger_glide_engraver (page 472), Fretboard_engraver (page 473), Note_heads_engraver (page 488), Note_name_engraver (page 488), Note_performer (page 489), Part_combine_engraver (page 491), Phrasing_slur_engraver (page 492), Slur_engraver (page 497), and Tab_note_heads_engraver (page 501).

1.2.56 note-grouping-event

Music event type note-grouping-event is in music objects of type NoteGroupingEvent (page 22).

Accepted by: Horizontal_bracket_engraver (page 477).

1.2.57 optional-material-event

Music event type optional-material-event is in music objects of type OptionalMaterialEvent (page 22).

Accepted by: Optional_material_bracket_engraver (page 489).

1.2.58 ottava-event

Music event type ottava-event is in music objects of type OttavaEvent (page 22).

Accepted by: Ottava_spanner_engraver (page 489).

1.2.59 page-break-event

Music event type page-break-event is in music objects of type PageBreakEvent (page 23).

Not accepted by any engraver or performer.

1.2.60 page-turn-event

Music event type page-turn-event is in music objects of type PageTurnEvent (page 23).

Not accepted by any engraver or performer.

1.2.61 part-combine-event

Music event type part-combine-event is in music objects of type PartCombineEvent (page 24).

Accepted by: Part_combine_engraver (page 491).

1.2.62 partial-event

Music event type partial-event is in music objects of type PartialEvent (page 24).

Accepted by: Timing_translator (page 505).

1.2.63 pedal-event

Music event type pedal-event is in music objects of type `SostenutoEvent` (page 36), `SustainEvent` (page 39), and `UnaCordaEvent` (page 44).

Not accepted by any engraver or performer.

1.2.64 percent-event

Music event type percent-event is in music objects of type `PercentEvent` (page 25).

Accepted by: `Percent_repeat_engraver` (page 492).

1.2.65 pes-or-flexa-event

Music event type pes-or-flexa-event is in music objects of type `PesOrFlexaEvent` (page 26).

Accepted by: `Vaticana_ligature_engraver` (page 507).

1.2.66 phrasing-slur-event

Music event type phrasing-slur-event is in music objects of type `PhrasingSlurEvent` (page 26).

Accepted by: `Phrasing_slur_engraver` (page 492).

1.2.67 rehearsal-mark-event

Music event type rehearsal-mark-event is in music objects of type `RehearsalMarkEvent` (page 28).

Accepted by: `Mark_tracking_translator` (page 483).

1.2.68 repeat-slash-event

Music event type repeat-slash-event is in music objects of type `RepeatSlashEvent` (page 29).

Accepted by: `Slash_repeat_engraver` (page 497).

1.2.69 repeat-tie-event

Music event type repeat-tie-event is in music objects of type `RepeatTieEvent` (page 30).

Accepted by: `Repeat_tie_engraver` (page 494).

1.2.70 rest-event

Music event type rest-event is in music objects of type `RestEvent` (page 30).

Accepted by: `Completion_rest_engraver` (page 464), `Figured_bass_engraver` (page 471), and `Rest_engraver` (page 495).

1.2.71 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 20), `NoteEvent` (page 21), `RepeatSlashEvent` (page 29), `RestEvent` (page 30), and `SkipEvent` (page 35).

Not accepted by any engraver or performer.

1.2.72 script-event

Music event type script-event is in music objects of type `ArticulationEvent` (page 3), `ScriptEvent` (page 31), and `TextScriptEvent` (page 40).

Not accepted by any engraver or performer.

1.2.73 section-event

Music event type section-event is in music objects of type SectionEvent (page 31).

Accepted by: Bar_engraver (page 453), and Divisio_engraver (page 467).

1.2.74 section-label-event

Music event type section-label-event is in music objects of type SectionLabelEvent (page 32).

Accepted by: Mark_tracking_translator (page 483).

1.2.75 segno-mark-event

Music event type segno-mark-event is in music objects of type SegnoMarkEvent (page 32).

Accepted by: Bar_engraver (page 453), and Mark_tracking_translator (page 483).

1.2.76 skip-event

Music event type skip-event is in music objects of type SkipEvent (page 35).

Not accepted by any engraver or performer.

1.2.77 slur-event

Music event type slur-event is in music objects of type SlurEvent (page 36).

Accepted by: Slur_engraver (page 497), and Slur_performer (page 498).

1.2.78 sostenuto-event

Music event type sostenuto-event is in music objects of type SostenutoEvent (page 36).

Accepted by: Piano_pedal_engraver (page 492), and Piano_pedal_performer (page 493).

1.2.79 spacing-section-event

Music event type spacing-section-event is in music objects of type SpacingSectionEvent (page 37).

Accepted by: Spacing_engraver (page 498).

1.2.80 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 470).

1.2.81 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 17), MeasureCounterEvent (page 19), MeasureSpannerEvent (page 19), OptionalMaterialEvent (page 22), PhrasingSlurEvent (page 26), SlurEvent (page 36), SostenutoEvent (page 36), SpanEvent (page 37), StaffHighlightEvent (page 37), StaffSpanEvent (page 38), SustainEvent (page 39), TextSpanEvent (page 40), TremoloSpanEvent (page 43), TrillSpanEvent (page 44), TupletSpanEvent (page 44), UnaCordaEvent (page 44), and VoltaSpanEvent (page 48).

Not accepted by any engraver or performer.

1.2.82 staff-highlight-event

Music event type staff-highlight-event is in music objects of type StaffHighlightEvent (page 37).

Accepted by: Staff_highlight_engraver (page 499).

1.2.83 staff-span-event

Music event type staff-span-event is in music objects of type StaffSpanEvent (page 38).

Accepted by: Staff_symbol_engraver (page 500).

1.2.84 StreamEvent

Music event type StreamEvent 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), BarCheckEvent (page 4), 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 16), LabelEvent (page 16), LaissezVibrerEvent (page 17), LigatureEvent (page 17), LineBreakEvent (page 17), LyricEvent (page 18), MeasureCounterEvent (page 19), MeasureSpannerEvent (page 19), MultiMeasureArticulationEvent (page 19), MultiMeasureRestEvent (page 20), MultiMeasureTextEvent (page 20), NoteEvent (page 21), NoteGroupingEvent (page 22), OptionalMaterialEvent (page 22), OttavaEvent (page 22), PageBreakEvent (page 23), PageTurnEvent (page 23), PartCombineEvent (page 24), PartialEvent (page 24), PercentEvent (page 25), PesOrFlexaEvent (page 26), PhrasingSlurEvent (page 26), RehearsalMarkEvent (page 28), RepeatSlashEvent (page 29), RepeatTieEvent (page 30), RestEvent (page 30), ScriptEvent (page 31), SectionEvent (page 31), SectionLabelEvent (page 32), SegnoMarkEvent (page 32), SkipEvent (page 35), SlurEvent (page 36), SostenutoEvent (page 36), SpacingSectionEvent (page 37), SpanEvent (page 37), StaffHighlightEvent (page 37), StaffSpanEvent (page 38), StringNumberEvent (page 38), StrokeFingerEvent (page 38), SustainEvent (page 39), TempoChangeEvent (page 39), TextMarkEvent (page 39), TextScriptEvent (page 40), TextSpanEvent (page 40), TieEvent (page 40), TimeSignatureEvent (page 41), TremoloEvent (page 42), TremoloSpanEvent (page 43), TrillSpanEvent (page 44), TupletSpanEvent (page 44), UnaCordaEvent (page 44), VoltaRepeatEndEvent (page 46), VoltaRepeatStartEvent (page 47), VoltaSpanEvent (page 48), and VowelTransitionEvent (page 49).

Not accepted by any engraver or performer.

1.2.85 string-number-event

Music event type string-number-event is in music objects of type StringNumberEvent (page 38).

Accepted by: Bend_spanner_engraver (page 459), Fretboard_engraver (page 473), and Tab_note_heads_engraver (page 501).

1.2.86 stroke-finger-event

Music event type stroke-finger-event is in music objects of type StrokeFingerEvent (page 38).

Not accepted by any engraver or performer.

1.2.87 structural-event

Music event type structural-event is in music objects of type AlternativeEvent (page 2), CodaMarkEvent (page 8), DalSegnoEvent (page 10), FineEvent (page 13), SectionEvent (page 31), SegnoMarkEvent (page 32), VoltaRepeatEndEvent (page 46), and VoltaRepeatStartEvent (page 47).

Not accepted by any engraver or performer.

1.2.88 sustain-event

Music event type sustain-event is in music objects of type SustainEvent (page 39).

Accepted by: Piano_pedal_engraver (page 492), and Piano_pedal_performer (page 493).

1.2.89 tempo-change-event

Music event type tempo-change-event is in music objects of type TempoChangeEvent (page 39).

Accepted by: Metronome_mark_engraver (page 485).

1.2.90 text-mark-event

Music event type text-mark-event is in music objects of type TextMarkEvent (page 39).

Accepted by: Text_mark_engraver (page 503).

1.2.91 text-script-event

Music event type text-script-event is in music objects of type TextScriptEvent (page 40).

Accepted by: Text_engraver (page 503).

1.2.92 text-span-event

Music event type text-span-event is in music objects of type TextSpanEvent (page 40).

Accepted by: Text_spanner_engraver (page 503).

1.2.93 tie-event

Music event type tie-event is in music objects of type TieEvent (page 40).

Accepted by: Drum_note_performer (page 468), Note_performer (page 489), Tie_engraver (page 503), and Tie_performer (page 504).

1.2.94 time-signature-event

Music event type time-signature-event is in music objects of type TimeSignatureEvent (page 41).

Accepted by: Time_signature_engraver (page 504), and Time_signature_performer (page 505).

1.2.95 tremolo-event

Music event type tremolo-event is in music objects of type TremoloEvent (page 42).

Accepted by: Stem_engraver (page 500).

1.2.96 tremolo-span-event

Music event type tremolo-span-event is in music objects of type TremoloSpanEvent (page 43).

Accepted by: Chord_tremolo_engraver (page 462).

1.2.97 trill-span-event

Music event type trill-span-event is in music objects of type TrillSpanEvent (page 44).

Accepted by: Trill_spanner_engraver (page 506).

1.2.98 tuplet-span-event

Music event type tuplet-span-event is in music objects of type TupletSpanEvent (page 44).

Accepted by: Tuplet_engraver (page 506).

1.2.99 una-corda-event

Music event type una-corda-event is in music objects of type UnaCordaEvent (page 44).

Accepted by: Piano_pedal_engraver (page 492), and Piano_pedal_performer (page 493).

1.2.100 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 467), Lyric_repeat_count_engraver (page 482), Repeat_acknowledge_engraver (page 494), and Signum_repetitionis_engraver (page 497).

1.2.101 volta-repeat-start-event

Music event type volta-repeat-start-event is in music objects of type VoltaRepeatStartEvent (page 47).

Accepted by: Divisio_engraver (page 467), and Repeat_acknowledge_engraver (page 494).

1.2.102 volta-span-event

Music event type volta-span-event is in music objects of type VoltaSpanEvent (page 48).

Accepted by: Volta_engraver (page 507).

1.2.103 vowel-transition-event

Music event type vowel-transition-event is in music objects of type VowelTransitionEvent (page 49).

Accepted by: Hyphen_engraver (page 477).

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 `\lyricsto` section.

associated-context-type (symbol)

Type of the context associated with this `\lyricsto` section.

augmented (boolean)

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

augmented-slash (boolean)

This figure is for an augmented figured bass (back-slashed number).

automatically-numbered (boolean)

Should a footnote be automatically numbered?

autosplit-end (boolean)

Duration of event was truncated by automatic splitting in `Completion_heads_engraver`.

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.

cautionary (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 (rational number)

Denominator of a ratio.

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.

`horizontal-direction` (direction)

This is RIGHT for `\textMark`, and LEFT for `\textEndMark`.

`id` (index or 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 of numbers)

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 (rational number)

Numerator of a ratio.

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 instantiated 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`.

`stroke-finger-digit` (non-negative, exact integer)

Digit for stroke finger.

`stroke-finger-text` (markup)

Markup expression to be printed for stroke finger.

`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 535), InstrumentName (page 618), SpanBarStub (page 695), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), and VerticalAlignment (page 742).

This context sets the following properties:

- Revert grob property extra-spacing-width in DynamicText (page 597),
- Set context property instrumentName to '()'.
 This property is used by the InstrumentName layout object.
- Set context property localAlterations to #f.
- Set context property localAlterations to '()'.
 This property is used by the Arpeggio layout object.
- Set context property shortInstrumentName to '()'.
 This property is used by the InstrumentName layout object.
- Set context property systemStartDelimiter to 'SystemStartBracket'.
 This property is used by the SystemStartBracket layout object.
- Set context property topLevelAlignment to #f.
- Set grob property extra-spacing-width in DynamicText (page 597), to #f.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Staff (page 309).

Context ChoirStaff can contain ChoirStaff (page 67), ChordNames (page 98), Devnull (page 110), DrumStaff (page 111), Dynamics (page 130), FiguredBass (page 135), FretBoards (page 136), GrandStaff (page 139), GregorianTranscriptionLyrics (page 141), GregorianTranscriptionStaff (page 144), KievanStaff (page 194), Lyrics (page 218), MensuralStaff (page 221), NoteNames (page 245), OneStaff (page 249), PetrucciStaff (page 250), PianoStaff (page 275), RhythmicStaff (page 277), Staff (page 309), StaffGroup (page 322), TabStaff (page 365), VaticanaLyrics (page 388), and VaticanaStaff (page 414).

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

Instrument_name_engraver (page 477)

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 618).

Output_property_engraver (page 490)
Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Span_arpeggio_engraver (page 498)
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 535).

Span_bar_stub_engraver (page 498)
Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s): SpanBarStub (page 695).

System_start_delimiter_engraver (page 501)
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 713), SystemStartBrace (page 714), SystemStartBracket (page 715), and SystemStartSquare (page 716).

Vertical_align_engraver (page 507)
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 698), and VerticalAlignment (page 742).

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 309).

This context creates the following layout object(s): BarLine (page 538), ChordSquare (page 564), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), GridChordName (page 612), PercentRepeat (page 667), PercentRepeatCounter (page 668), StaffSymbol (page 701), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), and VerticalAxisGroup (page 742).

This context sets the following properties:

- Set grob property font-size in BarLine (page 538), to 3.
- Set grob property hair-thickness in BarLine (page 538), to 2.
- Set grob property kern in BarLine (page 538), to 5.
- Set grob property line-positions in StaffSymbol (page 701), to :
'(-13.5 13.5)
- Set grob property thickness in StaffSymbol (page 701), to 2.
- Set grob property thickness in SystemStartBar (page 713), 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 451)

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 453)

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 742).

Bar_engraver (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)
 Notate volta-style repeats even when the repeat count is 1.

`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 return-count`
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Chord_square_engraver` (page 462)

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 564).

`Current_chord_text_engraver` (page 466)

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 52), and `note-event` (page 55),

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.

Double_percent_repeat_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Grid_chord_name_engraver (page 476)

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 612).

Output_property_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Percent_repeat_engraver (page 492)

Make whole measure repeats.

Music types accepted: percent-event (page 56),

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 667), and PercentRepeatCounter (page 668).

Staff_symbol_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

System_start_delimiter_engraver (page 501)

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 713), SystemStartBrace (page 714), SystemStartBracket (page 715), and SystemStartSquare (page 716).

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 283), and Timing (page 283).

This context creates the following layout object(s): BreakAlignGroup (page 554), BreakAlignment (page 555), CenteredBarNumberLineSpanner (page 562), CodaMark (page 570), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), GraceSpacing (page 611), JumpScript (page 621), LeftEdge (page 632), MetronomeMark (page 647), NonMusicalPaperColumn (page 655), PaperColumn (page 665), Parentheses (page 666), RehearsalMark (page 673), SectionLabel (page 681), SegnoMark (page 683), SpacingSpanner (page 693), StaffGrouper (page 698), TextMark (page 719), VerticalAlignment (page 742), VoltaBracket (page 745), and VoltaBracketSpanner (page 747).

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
measureStartNow
lastChord)

```

- Set context property associatedVoiceType to 'Voice.
- Set context property autoAccidentals to:

```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```
- 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-
- Set context property beamHalfMeasure to #t.
- Set context property breathMarkDefinitions to:

```
'((altcomma
  (text #<procedure musicglyph-markup (layout props glyph-name)>
    "scripts.raltcomma"))
 (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-maio (>)>
    (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.uupbow"))
(varcomma
  (text #<procedure musicglyph-markup (layout props glyph-name)>
    "scripts.rvarcomma")))

```

- 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 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 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 at
lily/translation-functions.scm:228:4 (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>.
- 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)
 (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 `"m"`.
- Set context property `noChordSymbol` to `"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 `printInitialRepeatBar` to `#t`.
- Set context property `printKeyCancellation` to `#t`.
- Set context property `printOctaveNames` to `#f`.
- Set context property `printPartCombineTexts` to `#t`.
- Set context property `printTrivialVoltaRepeats` to `#f`.
- Set context property `proportionalNotationDuration` to `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 at lily/translation-functions.scm:228:4 (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-axis . 1)
  (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)
  (side-axis . 1)
  (direction . 1))
(altcomma
  (script-stencil feta "laltcomma" . "raltcomma")
  (quantize-position . #t)
  (padding . 0.2)
  (avoid-slur . ignore)
  (side-axis . 1)
  (direction . 1))
(circulus
  (script-stencil feta "circulus" . "circulus")
  (side-relative-direction . -1)
  (avoid-slur . ignore)
  (padding . 0.2)
  (quantize-position . #t)
  (script-priority . -100)
  (side-axis . 1)
  (direction . 1))
(coda (script-stencil feta "coda" . "coda")
  (padding . 0.2)
  (avoid-slur . outside)
  (side-axis . 1)
  (direction . 1))
(comma (script-stencil feta "lcomma" . "rcomma")
  (quantize-position . #t)
  (padding . 0.2)
  (avoid-slur . ignore)
  (side-axis . 1)
  (direction . 1))
(downbow
  (script-stencil feta "ddownbow" . "udownbow")
  (padding . 0.2)
  (skyline-horizontal-padding . 0.2)
  (avoid-slur . around)
  (direction . 1))
```

```

(side-axis . 1)
(script-priority . 180))
(downmordent
  (script-stencil
    feta
    "downmordent"
    .
    "downmordent")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(downprall
  (script-stencil feta "downprall" . "downprall")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(espressivo
  (avoid-slur . around)
  (padding . 0.2)
  (script-stencil feta "espr" . "espr")
  (side-axis . 1)
  (side-relative-direction . -1))
(fermata
  (script-stencil feta "dfermata" . "ufermata")
  (padding . 0.4)
  (avoid-slur . around)
  (outside-staff-priority . 75)
  (script-priority . 175)
  (side-axis . 1)
  (direction . 1))
(flageolet
  (script-stencil feta "flageolet" . "flageolet")
  (padding . 0.2)
  (avoid-slur . around)
  (direction . 1)
  (side-axis . 1)
  (script-priority . 50))
(halfopen
  (avoid-slur . outside)
  (padding . 0.2)
  (script-stencil feta "halfopen" . "halfopen")
  (side-axis . 1)
  (direction . 1))
(halfopenvertical
  (avoid-slur . outside)
  (padding . 0.2)
  (script-stencil
    feta
    "halfopenvertical"
    .

```

```

    "halfopenvertical")
  (side-axis . 1)
  (direction . 1))
(haydnturn
  (script-stencil feta "haydnturn" . "haydnturn")
  (padding . 0.2)
  (avoid-slur . inside)
  (side-axis . 1)
  (direction . 1))
(henzelongfermata
  (script-stencil
    feta
    "dhenzelongfermata"
    .
    "uhenzelongfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (outside-staff-priority . 75)
  (script-priority . 175)
  (side-axis . 1)
  (direction . 1))
(henzeshortfermata
  (script-stencil
    feta
    "dhenzeshortfermata"
    .
    "uhenzeshortfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (outside-staff-priority . 75)
  (script-priority . 175)
  (side-axis . 1)
  (direction . 1))
(ictus (script-stencil feta "ictus" . "ictus")
  (side-relative-direction . -1)
  (quantize-position . #t)
  (avoid-slur . ignore)
  (padding . 0.2)
  (script-priority . -100)
  (side-axis . 1)
  (direction . -1))
(lheel (script-stencil feta "upedalheel" . "upedalheel")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . -1))
(lineprall
  (script-stencil feta "lineprall" . "lineprall")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))

```

```

(longfermata
  (script-stencil
    feta
    "dlongfermata"
    .
    "ulongfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (outside-staff-priority . 75)
  (script-priority . 175)
  (side-axis . 1)
  (direction . 1))
(ltoe (script-stencil feta "upedaltoe" . "upedaltoe")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . -1))
(marcato
  (script-stencil feta "dmarcato" . "umarcato")
  (padding . 0.2)
  (avoid-slur . inside)
  (quantize-position . #t)
  (side-axis . 1)
  (side-relative-direction . -1))
(mordent
  (script-stencil feta "mordent" . "mordent")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(open (avoid-slur . outside)
  (padding . 0.2)
  (script-stencil feta "open" . "open")
  (side-axis . 1)
  (direction . 1))
(outsidecomma
  (avoid-slur . around)
  (direction . 1)
  (padding . 0.2)
  (side-axis . 1)
  (script-stencil feta "lcomma" . "rcomma"))
(portato
  (script-stencil feta "uportato" . "dportato")
  (avoid-slur . around)
  (padding . 0.45)
  (side-axis . 1)
  (side-relative-direction . -1))
(prall (script-stencil feta "prall" . "prall")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))

```

```

(pralldown
  (script-stencil feta "pralldown" . "pralldown")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(prallmordent
  (script-stencil
    feta
    "prallmordent"
    .
    "prallmordent")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(prallprall
  (script-stencil feta "prallprall" . "prallprall")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(prallup
  (script-stencil feta "prallup" . "prallup")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(reverseturn
  (script-stencil
    feta
    "reverseturn"
    .
    "reverseturn")
  (padding . 0.2)
  (avoid-slur . inside)
  (side-axis . 1)
  (direction . 1))
(rheel (script-stencil feta "dpedalheel" . "dpedalheel")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(rtoe (script-stencil feta "dpedaltoe" . "dpedaltoe")
  (padding . 0.2)
  (avoid-slur . around)
  (side-axis . 1)
  (direction . 1))
(segno (script-stencil feta "segno" . "segno")
  (padding . 0.2)
  (avoid-slur . outside)
  (side-axis . 1))

```

```

        (direction . 1))
(semicirculus
  (script-stencil
    feta
    "dsemicirculus"
    .
    "dsemicirculus")
  (side-relative-direction . -1)
  (quantize-position . #t)
  (avoid-slur . ignore)
  (padding . 0.2)
  (script-priority . -100)
  (side-axis . 1)
  (direction . 1))
(shortfermata
  (script-stencil
    feta
    "dshortfermata"
    .
    "ushortfermata")
  (padding . 0.4)
  (avoid-slur . around)
  (outside-staff-priority . 75)
  (script-priority . 175)
  (side-axis . 1)
  (direction . 1))
(signumcongruentiae
  (script-stencil
    feta
    "dsignumcongruentiae"
    .
    "usignumcongruentiae")
  (padding . 0.2)
  (avoid-slur . outside)
  (side-axis . 1)
  (direction . 1))
(slashturn
  (script-stencil feta "slashturn" . "slashturn")
  (padding . 0.2)
  (avoid-slur . inside)
  (side-axis . 1)
  (direction . 1))
(snappizzicato
  (script-stencil
    feta
    "snappizzicato"
    .
    "snappizzicato")
  (padding . 0.2)
  (avoid-slur . outside)
  (side-axis . 1)
  (direction . 1))

```

```

(staccatissimo
  (avoid-slur . inside)
  (quantize-position . #t)
  (script-stencil
    feta
    "dstaccatissimo"
    .
    "ustaccatissimo")
  (padding . 0.2)
  (skyline-horizontal-padding . 0.1)
  (side-axis . 1)
  (side-relative-direction . -1)
  (toward-stem-shift . 1.0)
  (toward-stem-shift-in-column . 0.0))
(staccato
  (script-stencil feta "staccato" . "staccato")
  (side-axis . 1)
  (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)
  (side-axis . 1)
  (direction . 1))
(tenuto
  (script-stencil feta "tenuto" . "tenuto")
  (quantize-position . #t)
  (avoid-slur . inside)
  (padding . 0.2)
  (script-priority . -50)
  (side-axis . 1)
  (side-relative-direction . -1))
(trill (script-stencil feta "trill" . "trill")
  (direction . 1)
  (padding . 0.2)
  (avoid-slur . outside)
  (side-axis . 1)
  (script-priority . 150))
(turn (script-stencil feta "turn" . "turn")
  (avoid-slur . inside)
  (padding . 0.2)
  (side-axis . 1)
  (direction . 1))
(upbow (script-stencil feta "dupbow" . "uupbow")
  (avoid-slur . around)

```



```

        (padding . 0.2)
        (direction . 1)
        (side-axis . 1)
        (script-priority . 180))
(upmordent
 (script-stencil feta "upmordent" . "upmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(upprall
 (script-stencil feta "upprall" . "upprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(varcoda
 (script-stencil feta "varcoda" . "varcoda")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(varcomma
 (script-stencil feta "lvarcomma" . "rvarcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(verylongfermata
 (script-stencil
  feta
  "dverylongfermata"
  .
  "uverylongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(veryshortfermata
 (script-stencil
  feta
  "dveryshortfermata"
  .
  "uveryshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1))

```

```
(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 "/".
- 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 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 4))))
((9 . 4)
(beamExceptions (end (1/32 8 8 8 8 8 8 8 8))))

```
((12 . 4)
 (beamExceptions
  (end (1/32 8 8 8 8 8 8 8 8 8 8 8 8))))
((5 . 8) (beatStructure 3 2))
((8 . 8) (beatStructure 3 3 2)))
```

- Set 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 309).

Context `ChordGridScore` can contain `ChoirStaff` (page 67), `ChordGrid` (page 69), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Beam_collision_engraver` (page 457)

Help beams avoid colliding with notes and clefs in other voices.

`Break_align_engraver` (page 460)

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 554), `BreakAlignment` (page 555), and `LeftEdge` (page 632).

`Centered_bar_number_align_engraver` (page 461)

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 562).

`Concurrent_hairpin_engraver` (page 465)

Collect concurrent hairpins.

`Footnote_engraver` (page 473)

Create footnote texts.

This engraver creates the following layout object(s): `Footnote` (page 606).

`Grace_spacing_engraver` (page 475)

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 611).

Jump_engraver (page 478)

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 499), also needs to be there so that marks appear at the intended Y location.

Music types accepted: *ad-hoc-jump-event* (page 49), *dal-segno-event* (page 51), and *fine-event* (page 52),

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 621).

`Mark_engraver` (page 482)

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 499), 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 483). 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 570), `RehearsalMark` (page 673), `SectionLabel` (page 681), and `SegnoMark` (page 683).

`Mark_tracking_translator` (page 483)

This translator chooses which marks `Mark_engraver` should engrave.

Music types accepted: `ad-hoc-mark-event` (page 49), `coda-mark-event` (page 51), `rehearsal-mark-event` (page 56), `section-label-event` (page 57), and `segno-mark-event` (page 57),

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 485)

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 Staff_collecting_engraver (page 499).

Music types accepted: tempo-change-event (page 59),

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 647).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Paper_column_engraver` (page 490)

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 50), and `label-event` (page 53),

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 655), and `PaperColumn` (page 665).

`Parenthesis_engraver` (page 491)

Parenthesize objects whose `parenthesize` property is `#t`.

This engraver creates the following layout object(s): `Parentheses` (page 666).

`Repeat_acknowledge_engraver` (page 494)

This translator adds entries to `repeatCommands` for events generated by `\\repeat volta`.

Music types accepted: `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 return-count`

End a repeated section. `return-count` is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 496)

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 575), and `ControlPolygon` (page 576).

`Spacing_engraver` (page 498)

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted: `spacing-section-event` (page 57),

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` (non-negative exact rational or `+inf.0`)

Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s): `SpacingSpanner` (page 693).

`Spanner_tracking_engraver` (page 499)

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 499)

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 500)

This engraver ensures that stanza numbers are neatly aligned.

`Text_mark_engraver` (page 503)

Engraves arbitrary textual marks.

Music types accepted: `text-mark-event` (page 59),

Properties (read)

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): TextMark (page 719).

Timing_translator (page 505)

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 49), bar-check-event (page 50), bar-event (page 50), fine-event (page 52), and partial-event (page 55),

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

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 (positive exact rational or +inf.0)

The musical length of the current measure.

measurePosition (moment)

The current point within the measure.

timeSignatureFraction (simple time signature)

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 first volta number for the current \alternative element. Not set outside of alternatives.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

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 (positive exact rational or +inf.0)

The musical length of the current measure.

measurePosition (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`timeSignatureFraction` (simple time signature)

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

`Tweak_engraver` (page 507)

Read the tweaks property from the originating event, and set properties.

`Vertical_align_engraver` (page 507)

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 698), and `VerticalAlignment` (page 742).

`Volta_engraver` (page 507)

Make volta brackets.

Music types accepted: `dal-segno-event` (page 51), `fine-event` (page 52), and `volta-span-event` (page 60),

Properties (read)

`currentCommandColumn` (graphical (layout) object)

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

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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.

This engraver creates the following layout object(s): `VoltaBracket` (page 745), and `VoltaBracketSpanner` (page 747).

2.1.4 ChordNames

Typesets chord names.

This context also accepts commands for the following context(s): `Staff` (page 309).

This context creates the following layout object(s): `ChordName` (page 563), `StaffSpacing` (page 700), and `VerticalAxisGroup` (page 742).

This context sets the following properties:

- Set grob property `font-size` in `Parentheses` (page 666), to 1.5.
- Set grob property `nonstaff-nonstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 0.5.
- Set grob property `nonstaff-relatedstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 0.5.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), 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 451)

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 453)

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 742).

`Chord_name_engraver` (page 461)

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 563).

`Current_chord_text_engraver` (page 466)

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 52), and `note-event` (page 55),

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.

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Separating_line_group_engraver (page 496)

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 700).

2.1.5 CueVoice

A voice context used to render notes of a reduced size, intended primarily for adding cue notes to a staff. Usually left to be created implicitly.

This context also accepts commands for the following context(s): Voice (page 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), FingerGlideSpanner (page 601), Fingering (page 603), Flag (page 605), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), LigatureBracket (page 634), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), and VoiceFollower (page 744).

This context sets the following properties:

- Set context property `fontSize` to -4.
- Set grob property `beam-thickness` in Beam (page 549), to 0.35.
- Set grob property `beam-thickness` in StemTremolo (page 705), to 0.35.
- Set grob property `ignore-ambitus` in NoteHead (page 658), to #t.
- Set grob property `length-fraction` in Beam (page 549), to 0.6299605249474366.
- Set grob property `length-fraction` in Stem (page 703), 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 452)

Generate an Arpeggio symbol.

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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto_beam_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Bend_engraver (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

Breathing_sign_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

Chord_tremolo_engraver (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster_spanner_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double_percent_repeat_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Dynamic_align_engraver (page 469)

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 595).

Dynamic_engraver (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 49), break-dynamic-span-event (page 50), and span-dynamic-event (page 57),

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 597), DynamicTextSpanner (page 599), and Hairpin (page 614).

Finger_glide_engraver (page 472)

Engraver to print a line between two Fingering, StringNumber or StrokeFinger grobs.

Music types accepted: note-event (page 55),

This engraver creates the following layout object(s): FingerGlideSpanner (page 601).

Fingering_engraver (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

Font_size_engraver (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

Properties (read)

`glissandoMap` (list)

A map in the form of `'((source1 . target1) (source2 . target2) (source . 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 610).

Grace_auto_beam_engraver (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

Grace_beam_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace_engraver (page 475)

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 476)

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 478)

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 619).

Laissez_vibrer_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Ligature_bracket_engraver (page 481)

Handle Ligature_events by engraving Ligature brackets.

Music types accepted: ligature-event (page 53),

This engraver creates the following layout object(s): LigatureBracket (page 634).

Multi_measure_rest_engraver (page 487)

Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54),

multi-measure-rest-event (page 54), and multi-measure-text-event (page 54),

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 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), and MultiMeasureRestText (page 653).

New_fingering_engraver (page 487)

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 603), Script (page 679), StringNumber (page 706), and StrokeFinger (page 708).

Note_head_line_engraver (page 488)

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 744).

`Note_heads_engraver` (page 488)

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

`Repeat_tie_engraver` (page 494)

Create repeat ties.

Music types accepted: `repeat-tie-event` (page 56),

This engraver creates the following layout object(s): `RepeatTie` (page 676), and `RepeatTieColumn` (page 677).

`Rest_engraver` (page 495)

Engrave rests.

Music types accepted: `rest-event` (page 56),

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 678).

`Rhythmic_column_engraver` (page 495)

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): `NoteColumn` (page 657).

`Script_column_engraver` (page 495)

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 680).

`Script_engraver` (page 496)

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 50),

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 679).

`Slash_repeat_engraver` (page 497)

Make beat repeats.

Music types accepted: `repeat-slash-event` (page 56),

This engraver creates the following layout object(s): `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

`Slur_engraver` (page 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

`Spanner_break_forbid_engraver` (page 499)

Forbid breaks in certain spanners.

`Stem_engraver` (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and

`TieColumn` (page 727).

`Trill_spanner_engraver` (page 506)

Create trill spanners.

Music types accepted: `trill-span-event` (page 60),

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 734).

`Tuplet_engraver` (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

2.1.6 Devnull

Silently discard all musical information given to this context.

This context also accepts commands for the following context(s): `Staff` (page 309), and `Voice` (page 439).

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. Can contain DrumVoice.

This context also accepts commands for the following context(s): Staff (page 309).

This context creates the following layout object(s): BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), BreathingSign (page 557), CaesuraScript (page 559), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SostenuatoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedalLineSpanner (page 711), TimeSignature (page 727), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

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")
  (-1 . "8")
  (-2 . "15")
  (-3 . "22")
  (-4 . "29"))
```
- Set context property shortInstrumentName to '().
- Set grob property staff-padding in Script (page 679), 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 120).

Context DrumStaff can contain CueVoice (page 100), DrumVoice (page 120), and NullVoice (page 247).

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

Alteration_glyph_engraver (page 451)

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 453)

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 742).

Bar_engraver (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar (boolean)`
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats (boolean)`
 Notate volta-style repeats even when the repeat count is 1.

`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 return-count`
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544),

`BassFigureAlignment` (page 544), `BassFigureBracket` (page 546),

`BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

Horizontal_script_engraver (page 477)

Aligns Script horizontally

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.

Instrument_name_engraver (page 477)

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 618).

Ledger_line_engraver (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

Merge_mmrest_numbers_engraver (page 485)

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 488)

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 680).

Optional_material_bracket_engraver (page 489)

Notate in-staff brackets for optional material.

- Music types accepted: optional-material-event (page 55),
 This engraver creates the following layout object(s): OptionalMaterialBracket (page 661).
- Output_property_engraver (page 490)
 Apply a procedure to any grob acknowledged.
 Music types accepted: apply-output-event (page 49),
- Piano_pedal_align_engraver (page 492)
 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 692), SustainPedalLineSpanner (page 711), and UnaCordaPedalLineSpanner (page 740).
- Pure_from_neighbor_engraver (page 494)
 Coordinates items that get their pure heights from their neighbors.
- Rest_collision_engraver (page 495)
 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 679).
- Script_row_engraver (page 496)
 Determine order in horizontal side position elements.
 This engraver creates the following layout object(s): ScriptRow (page 681).
- Separating_line_group_engraver (page 496)
 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 700).
- Skip_typesetting_engraver (page 497)
 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 696).

`Staff_collecting_engraver` (page 499)

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 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Time_signature_engraver` (page 504)

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

2.1.8 DrumVoice

A voice on a percussion staff.

This context also accepts commands for the following context(s): `Voice` (page 439).

This context creates the following layout object(s): `Beam` (page 549), `BendAfter` (page 551), `BreathingSign` (page 557), `CombineTextScript` (page 572), `Dots` (page 588), `DoublePercentRepeat` (page 590), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `FingerGlideSpanner` (page 601), `Flag` (page 605), `Hairpin` (page 614), `InstrumentSwitch` (page 619), `LaissezVibrerTie` (page 629), `LaissezVibrerTieColumn` (page 630), `MultiMeasureRest` (page 649),

MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), and TupletNumber (page 737).

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 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Bend_engraver` (page 459)

Create fall spanners.

Music types accepted: `bend-after-event` (page 50),

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 551).

`Breathing_sign_engraver` (page 460)

Notate breath marks.

Music types accepted: `breathing-event` (page 51),

Properties (read)

`breathMarkType` (symbol)

The type of `BreathingSign` to create at `\breathe`.

This engraver creates the following layout object(s): `BreathingSign` (page 557).

`Chord_tremolo_engraver` (page 462)

Generate beams for tremolo repeats.

Music types accepted: `tremolo-span-event` (page 59),

This engraver creates the following layout object(s): `Beam` (page 549).

`Dots_engraver` (page 468)

Create `Dots` (page 588), objects for `rhythmic-head-interface` (page 813)s.

This engraver creates the following layout object(s): `Dots` (page 588).

`Double_percent_repeat_engraver` (page 468)

Make double measure repeats.

Music types accepted: `double-percent-event` (page 51),

Properties (read)

`countPercentRepeats` (boolean)

If set, produce counters for percent repeats.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`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 590), and `DoublePercentRepeatCounter` (page 591).

`Drum_notes_engraver` (page 468)

Generate drum note heads.

Music types accepted: `note-event` (page 55),

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 658), and `Script` (page 679).

`Dynamic_align_engraver` (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49),

`break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two Fingering, `StringNumber` or `StrokeFinger` grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Font_size_engraver` (page 472)

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 473)

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.

`Grace_auto_beam_engraver` (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to #t then beams are generated automatically. If set to #f, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

`Grace_beam_engraver` (page 475)

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 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Grace_engraver` (page 475)

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 476)

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 478)

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 619).

`Laissez_vibrer_engraver` (page 481)

Create `laissez vibrer` items.

Music types accepted: `laissez-vibrer-event` (page 53),

This engraver creates the following layout object(s): `LaissezVibrerTie` (page 629), and `LaissezVibrerTieColumn` (page 630).

`Multi_measure_rest_engraver` (page 487)

Engrave multi-measure rests that are produced with ‘R’. It reads `measureStartNow` and `internalBarNumber` to determine what number to print over the `MultiMeasureRest` (page 649).

Music types accepted: `multi-measure-articulation-event` (page 54), `multi-measure-rest-event` (page 54), and `multi-measure-text-event` (page 54),
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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

Percent_repeat_engraver (page 492)

Make whole measure repeats.

Music types accepted: percent-event (page 56),

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 667), and PercentRepeatCounter (page 668).

Phrasing_slur_engraver (page 492)

Print phrasing slurs. Similar to Slur_engraver (page 497).

Music types accepted: note-event (page 55), and phrasing-slur-event (page 56),

This engraver creates the following layout object(s): PhrasingSlur (page 670).

Pitched_trill_engraver (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), and TrillPitchParentheses (page 733).

Repeat_tie_engraver (page 494)

Create repeat ties.

Music types accepted: repeat-tie-event (page 56),

This engraver creates the following layout object(s): RepeatTie (page 676), and RepeatTieColumn (page 677).

Rest_engraver (page 495)

Engrave rests.

Music types accepted: rest-event (page 56),

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 678).

Rhythmic_column_engraver (page 495)

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

Script_column_engraver (page 495)

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 680).

Script_engraver (page 496)

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 50),

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 679).

Slash_repeat_engraver (page 497)

Make beat repeats.

Music types accepted: `repeat-slash-event` (page 56),

This engraver creates the following layout object(s): `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

Slur_engraver (page 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

Spanner_break_forbid_engraver (page 499)

Forbid breaks in certain spanners.

Stem_engraver (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), Stem (page 703), StemStub (page 705), and StemTremolo (page 705).

Text_engraver (page 503)

Create text scripts.

Music types accepted: text-script-event (page 59),

This engraver creates the following layout object(s): TextScript (page 721).

Text_spanner_engraver (page 503)

Create text spanner from an event.

Music types accepted: text-span-event (page 59),

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 723).

Tie_engraver (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 59),

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 725), and

TieColumn (page 727).

Trill_spanner_engraver (page 506)

Create trill spanners.

Music types accepted: trill-span-event (page 60),

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 734).

Tuplet_engraver (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 60),

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 736), and `TupletNumber` (page 737).

2.1.9 Dynamics

Holds a single line of dynamics centered between the staves surrounding this context.

This context also accepts commands for the following context(s): `Staff` (page 309), and `Voice` (page 439).

This context creates the following layout object(s): `BarLine` (page 538), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `Hairpin` (page 614), `PianoPedalBracket` (page 672), `Script` (page 679), `SostenutoPedal` (page 690), `SustainPedal` (page 710), `TextScript` (page 721), `TextSpanner` (page 723), `UnaCordaPedal` (page 739), and `VerticalAxisGroup` (page 742).

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 721), to `'italic`.
- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 742), to :
'((basic-distance . 5) (padding . 0.5))
- Set grob property `outside-staff-priority` in `DynamicLineSpanner` (page 595), to `#f`.
- Set grob property `outside-staff-priority` in `DynamicText` (page 597), to `#f`.
- Set grob property `outside-staff-priority` in `Hairpin` (page 614), to `#f`.
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), to `0`.
- Set grob property `Y-offset` in `DynamicLineSpanner` (page 595), 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 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)
 Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Dynamic_align_engraver` (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 49), break-dynamic-span-event (page 50), and span-dynamic-event (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_engraver` (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

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 672), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `UnaCordaPedal` (page 739).

`Script_engraver` (page 496)

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 50),

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 679).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

2.1.10 FiguredBass

The context in which `BassFigure` grobs are created from input entered in `\figuremode` mode.

This context creates the following layout object(s): `BassFigure` (page 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), `BassFigureLine` (page 548), `StaffSpacing` (page 700), and `VerticalAxisGroup` (page 742).

This context sets the following properties:

- Set grob property `nonstaff-nonstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 0.5.
- Set grob property `nonstaff-relatedstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 0.5.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), 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 453)

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 742).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Separating_line_group_engraver` (page 496)

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 700).

2.1.11 FretBoards

A context for displaying fret diagrams.

This context also accepts commands for the following context(s): `Staff` (page 309).

This context creates the following layout object(s): `FretBoard` (page 608), `InstrumentName` (page 618), `StaffSpacing` (page 700), and `VerticalAxisGroup` (page 742).

This context sets the following properties:

- Set context property `handleNegativeFrets` to 'recalculate.
- Set context property `instrumentName` to '().
- Set context property `predefinedDiagramTable` to #<hash-table>.
- 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 453)

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 742).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Fretboard_engraver` (page 473)

Generate fret diagram from one or more events of type `NoteEvent`.

Music types accepted: `fingering-event` (page 52), `note-event` (page 55), and `string-number-event` (page 58),

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 608).

`Instrument_name_engraver` (page 477)

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 618).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Separating_line_group_engraver` (page 496)

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 700).

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 283).

Context Global can contain ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

2.1.13 GrandStaff

Connect staves vertically by adding a brace on the left side. The bar lines of the contained staves are connected vertically, too.

This context creates the following layout object(s): Arpeggio (page 535), InstrumentName (page 618), SpanBar (page 694), SpanBarStub (page 695), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), and VerticalAlignment (page 742).

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 597), to `#f`.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Staff (page 309).

Context GrandStaff can contain ChoirStaff (page 67), ChordNames (page 98), Devnull (page 110), DrumStaff (page 111), Dynamics (page 130), FiguredBass (page 135), FretBoards (page 136), GrandStaff (page 139), GregorianTranscriptionLyrics (page 141), GregorianTranscriptionStaff (page 144), KievanStaff (page 194), Lyrics (page 218), MensuralStaff (page 221), NoteNames (page 245), OneStaff (page 249), PetrucciStaff (page 250), PianoStaff (page 275), RhythmicStaff (page 277), Staff (page 309), StaffGroup (page 322), TabStaff (page 365), VaticanaLyrics (page 388), and VaticanaStaff (page 414).

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

`Instrument_name_engraver` (page 477)

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 618).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Span_arpeggio_engraver` (page 498)

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 535).

`Span_bar_engraver` (page 498)

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 694).

`Span_bar_stub_engraver` (page 498)

Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s): `SpanBarStub` (page 695).

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Vertical_align_engraver` (page 507)

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 698), and `VerticalAlignment` (page 742).

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 218).

This context creates the following layout object(s): `InstrumentName` (page 618), `LyricExtender` (page 635), `LyricHyphen` (page 636), `LyricRepeatCount` (page 637), `LyricSpace` (page 639), `LyricText` (page 640), `StanzaNumber` (page 702), `VerticalAxisGroup` (page 742), and `VowelTransition` (page 748).

This context sets the following properties:

- Set context property `instrumentName` to `'()`.
- Set context property `lyricRepeatCountFormatter` to `#<procedure at lily/translation-functions.scm:214:4 (context repeat-count)>`.
- Set context property `searchForVoice` to `#f`.
- Set context property `shortInstrumentName` to `'()`.
- Set grob property `bar-extent` in `BarLine` (page 538), to :
`'(-0.05 . 0.05)`
- Set grob property `font-size` in `InstrumentName` (page 618), to 1.0.
- Set grob property `nonstaff-nonstaff-spacing` in `VerticalAxisGroup` (page 742), to :
`'((basic-distance . 0)
 (minimum-distance . 2.8)
 (padding . 0.2)
 (stretchability . 0))`
- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 742), to :
`'((basic-distance . 5.5)
 (padding . 0.5)
 (stretchability . 1))`
- Set grob property `nonstaff-unrelatedstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 1.5.
- Set grob property `parent-alignment-X` in `LyricRepeatCount` (page 637), to 1.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `self-alignment-Y` in `InstrumentName` (page 618), to `#f`.
- Set grob property `short-bar-extent` in `BarLine` (page 538), to :
`'(-0.05 . 0.05)`
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), 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 453)

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 742).

`Extender_engraver` (page 471)

Create lyric extenders.

Music types accepted: `completize-extender-event` (page 51), and `extender-event` (page 52),

Properties (read)

`extendersOverRests` (boolean)

Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): `LyricExtender` (page 635).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Hyphen_engraver` (page 477)

Create lyric hyphens, vowel transitions and distance constraints between words.

Music types accepted: `hyphen-event` (page 53), and `vowel-transition-event` (page 60),

This engraver creates the following layout object(s): `LyricHyphen` (page 636), `LyricSpace` (page 639), and `VowelTransition` (page 748).

`Instrument_name_engraver` (page 477)

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 618).

Lyric_engraver (page 481)

Engrave text for lyrics.

Music types accepted: lyric-event (page 53),

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 640).

Lyric_repeat_count_engraver (page 482)

Create repeat counts within lyrics for modern transcriptions of Gregorian chant.

Music types accepted: volta-repeat-end-event (page 60),

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 637).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Stanza_number_engraver (page 500)

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 702).

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 309).

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), Divisio (page 585), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SostenuitoPedal (page 690), SostenuitoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

This context sets the following properties:

- Set context property `autoAccidentals` to:
`'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)`
- 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 `printTrivialVoltaRepeats` to `#t`.
- 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 557), to `item::extra-spacing-height-including-staff`.
- Set grob property `extra-spacing-width` in `BreathingSign` (page 557), 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 157).

Context `GregorianTranscriptionStaff` can contain `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), and `NullVoice` (page 247).

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

`Accidental_engraver` (page 450)

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 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), and `AccidentalSuggestion` (page 529).

`Alteration_glyph_engraver` (page 451)

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 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)
 Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Divisio_engraver` (page 467)

Create divisions: chant notation for points of breathing or caesura.

Music types accepted: `caesura-event` (page 51), `fine-event` (page 52), `section-event` (page 57), `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 585).

`Dot_column_engraver (page 467)`

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 588).

`Figured_bass_engraver (page 471)`

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

`Horizontal_script_engraver` (page 477)

Aligns Script horizontally

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.

`Instrument_name_engraver` (page 477)

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 618).

Key_engraver (page 479)

Engrave a key signature.

Music types accepted: key-change-event (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

Output_property_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Piano_pedal_align_engraver (page 492)

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 692), SustainPedalLineSpanner (page 711), and UnaCordaPedalLineSpanner (page 740).

Piano_pedal_engraver (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 57), sustain-event (page 59), and una-corda-event (page 60),

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 672), SostenutoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 495)

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 679).

`Script_row_engraver` (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): `ScriptRow` (page 681).

`Separating_line_group_engraver` (page 496)

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 700).

`Skip_typesetting_engraver` (page 497)

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 696).

`Staff_collecting_engraver` (page 499)

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 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

2.1.16 GregorianTranscriptionVoice

A voice for notating Gregorian chant in modern style.

This context also accepts commands for the following context(s): `Voice` (page 439).

This context creates the following layout object(s): `Arpeggio` (page 535), `Beam` (page 549), `BendAfter` (page 551), `BreathingSign` (page 557), `ClusterSpanner` (page 569), `ClusterSpannerBeacon` (page 570), `CombineTextScript` (page 572), `Dots` (page 588), `DoublePercentRepeat` (page 590), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `Episema` (page 600), `FingerGlideSpanner` (page 601), `Fingering` (page 603), `Glissando` (page 610), `Hairpin` (page 614), `InstrumentSwitch` (page 619), `LaissezVibrerTie` (page 629), `LaissezVibrerTieColumn` (page 630), `MultiMeasureRest` (page 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), `MultiMeasureRestText` (page 653), `NoteColumn` (page 657), `NoteHead` (page 658), `NoteSpacing` (page 660), `PercentRepeat` (page 667), `PercentRepeatCounter` (page 668), `PhrasingSlur` (page 670), `RepeatSlash` (page 675), `RepeatTie` (page 676), `RepeatTieColumn` (page 677), `Rest` (page 678), `Script` (page 679), `ScriptColumn` (page 680), `Slur` (page 688), `StringNumber` (page 706), `StrokeFinger` (page 708), `TextScript` (page 721), `TextSpanner` (page 723), `Tie` (page 725), `TieColumn` (page 727), `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), `TrillPitchParentheses` (page 733), `TrillSpanner` (page 734), `TupletBracket` (page 736), `TupletNumber` (page 737), and `VoiceFollower` (page 744).

This context sets the following properties:

- Set context property `autoBeaming` to `#f`.

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

This context cannot contain other contexts.

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

`Arpeggio_engraver` (page 452)

Generate an Arpeggio symbol.

Music types accepted: `arpeggio-event` (page 49),

This engraver creates the following layout object(s): `Arpeggio` (page 535).

`Auto_beam_engraver` (page 452)

Generate beams based on measure characteristics and observed Stems. Uses `beatBase`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through `Stem_engraver` (page 500), properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted: `beam-forbid-event` (page 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

`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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Bend_engraver (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

Breathing_sign_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

`breathMarkType` (symbol)

The type of `BreathingSign` to create at `\breathe`.

This engraver creates the following layout object(s): `BreathingSign` (page 557).

`Chord_tremolo_engraver` (page 462)

Generate beams for tremolo repeats.

Music types accepted: `tremolo-span-event` (page 59),

This engraver creates the following layout object(s): `Beam` (page 549).

`Cluster_spanner_engraver` (page 463)

Engrave a cluster using `Spanner` notation.

Music types accepted: `cluster-note-event` (page 51),

This engraver creates the following layout object(s): `ClusterSpanner` (page 569), and `ClusterSpannerBeacon` (page 570).

`Dots_engraver` (page 468)

Create `Dots` (page 588), objects for `rhythmic-head-interface` (page 813)s.

This engraver creates the following layout object(s): `Dots` (page 588).

`Double_percent_repeat_engraver` (page 468)

Make double measure repeats.

Music types accepted: `double-percent-event` (page 51),

Properties (read)

`countPercentRepeats` (boolean)

If set, produce counters for percent repeats.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`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 590), and `DoublePercentRepeatCounter` (page 591).

`Dynamic_align_engraver` (page 469)

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 595).

Dynamic_engraver (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49),
`break-dynamic-span-event` (page 50), and `span-dynamic-event`
 (page 57),

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 597),
`DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

Episema_engraver (page 471)

Create an *Editio Vaticana*-style episema line.

Music types accepted: `episema-event` (page 52),

This engraver creates the following layout object(s): `Episema` (page 600).

Finger_glide_engraver (page 472)

Engraver to print a line between two `Fingering`, `StringNumber` or `StrokeFinger`
 grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner`
 (page 601).

Fingering_engraver (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

Font_size_engraver (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: glissando-event (page 52),

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 610).

Grace_auto_beam_engraver (page 474)

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 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): Beam (page 549).

Grace_beam_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace_engraver (page 475)

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 476)

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 478)

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 619).

Laissez_vibrer_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Multi_measure_rest_engraver (page 487)

Engrave multi-measure rests that are produced with 'R'. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54),

multi-measure-rest-event (page 54), and multi-measure-text-event (page 54),

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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

`New_fingering_engraver` (page 487)

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 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

`Note_head_line_engraver` (page 488)

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 744).

`Note_heads_engraver` (page 488)

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

Repeat_tie_engraver (page 494)

Create repeat ties.

Music types accepted: repeat-tie-event (page 56),

This engraver creates the following layout object(s): RepeatTie (page 676), and RepeatTieColumn (page 677).

Rest_engraver (page 495)

Engrave rests.

Music types accepted: rest-event (page 56),

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 678).

Rhythmic_column_engraver (page 495)

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

Script_column_engraver (page 495)

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 680).

Script_engraver (page 496)

Handle note scripted articulations.

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

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 679).

Slash_repeat_engraver (page 497)

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

Slur_engraver (page 497)

Build slur grobs from slur events.

Music types accepted: note-event (page 55), and slur-event (page 57),

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 688).

Spanner_break_forbid_engraver (page 499)

Forbid breaks in certain spanners.

Text_engraver (page 503)

Create text scripts.

Music types accepted: text-script-event (page 59),

This engraver creates the following layout object(s): TextScript (page 721).

Text_spanner_engraver (page 503)

Create text spanner from an event.

Music types accepted: text-span-event (page 59),

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 723).

Tie_engraver (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 59),

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 725), and

TieColumn (page 727).

Trill_spanner_engraver (page 506)

Create trill spanners.

Music types accepted: trill-span-event (page 60),

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 734).

Tuplet_engraver (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 60),

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 736), and `TupletNumber` (page 737).

2.1.17 InternalGregorianStaff

A kind of `Staff` with settings shared by multiple variants of Gregorian chant notation.

This context creates the following layout object(s): `Accidental` (page 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), `AccidentalSuggestion` (page 529), `BarLine` (page 538), `BassFigure` (page 544), `BassFigureAlignment` (page 544), `BassFigureAlignmentPositioning` (page 545), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), `BassFigureLine` (page 548), `Clef` (page 565), `ClefModifier` (page 568), `CueClef` (page 577), `CueEndClef` (page 580), `Divisio` (page 585), `DotColumn` (page 588), `FingeringColumn` (page 605), `InstrumentName` (page 618), `KeyCancellation` (page 622), `KeySignature` (page 625), `LedgerLineSpanner` (page 631), `NoteCollision` (page 656), `OptionalMaterialBracket` (page 661), `OttavaBracket` (page 664), `PianoPedalBracket` (page 672), `RestCollision` (page 679), `ScriptColumn` (page 680), `ScriptRow` (page 681), `SostenutoPedal` (page 690), `SostenutoPedalLineSpanner` (page 692), `StaffEllipsis` (page 696), `StaffHighlight` (page 699), `StaffSpacing` (page 700), `StaffSymbol` (page 701), `SustainPedal` (page 710), `SustainPedalLineSpanner` (page 711), `TimeSignature` (page 727), `UnaCordaPedal` (page 739), `UnaCordaPedalLineSpanner` (page 740), and `VerticalAxisGroup` (page 742).

This context sets the following properties:

- Set context property `autoAccidentals` to:
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
- 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 `printTrivialVoltaRepeats` to `#t`.
- 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 557), to `item::extra-spacing-height-including-staff`.
- Set grob property `extra-spacing-width` in `BreathingSign` (page 557), to :
`'(-1.0 . 0.0)`

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

Context `InternalGregorianStaff` can contain `CueVoice` (page 100), and `NullVoice` (page 247).

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

`Accidental_engraver` (page 450)

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 526), *AccidentalCautionary* (page 527), *AccidentalPlacement* (page 528), and *AccidentalSuggestion* (page 529).

Alteration_glyph_engraver (page 451)

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 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Divisio_engraver` (page 467)

Create divisiones: chant notation for points of breathing or caesura.

Music types accepted: `caesura-event` (page 51), `fine-event` (page 52), `section-event` (page 57), `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 585).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

`Horizontal_script_engraver` (page 477)

Aligns Script horizontally

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.

`Instrument_name_engraver` (page 477)

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 618).

`Key_engraver` (page 479)

Engrave a key signature.

Music types accepted: `key-change-event` (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

`Piano_pedal_engraver` (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

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 672), SostenuatoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 495)

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 679).

Script_row_engraver (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Separating_line_group_engraver (page 496)

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 700).

Skip_typesetting_engraver (page 497)

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 696).

Staff_collecting_engraver (page 499)

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 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff_symbol_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

Time_signature_engraver (page 504)

Create a TimeSignature (page 727), whenever timeSignatureFraction changes.

Music types accepted: time-signature-event (page 59),

Properties (read)

initialTimeSignatureVisibility (vector)

break visibility for the initial time signature.

partialBusy (boolean)

Signal that \partial acts at the current timestep.

timeSignatureFraction (simple time signature)

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 727).

2.1.18 InternalMensuralStaff

An kind of Staff with settings shared by multiple variants of mensural notation.

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), BreathingSign (page 557), CaesuraScript (page 559), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), Custos (page 583), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SignumRepetitionis (page 685), SostenuatoPedal (page 690), SostenuatoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), TimeSignature (page 727), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

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 at lily/music-functions.scm:1714:0 (context pitch barnum)>)
- Set context property `autoCautionaries` to '().
- Set context property `caesuraType` to:
'((bar-line . "|"))
- 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 `fineSegnoBarType` to "|.S".
- Set context property `fineStartRepeatSegnoBarType` to "|.S".
- 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 `printKeyCancellation` to #f.
- 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 `neutral-direction` in `Custos` (page 583), to -1.
- Set grob property `neutral-position` in `Custos` (page 583), to 3.
- Set grob property `style` in `Custos` (page 583), to 'mensural.
- Set grob property `style` in `TimeSignature` (page 727), to 'mensural.

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

Context `InternalMensuralStaff` can contain `CueVoice` (page 100), and `NullVoice` (page 247).

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

`Accidental_engraver` (page 450)

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 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), and AccidentalSuggestion (page 529).

Alteration_glyph_engraver (page 451)

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 453)

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 742).

Bar_engraver (page 453)

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 49), caesura-event (page 51), coda-mark-event (page 51), dal-segno-event (page 51), fine-event (page 52), section-event (page 57), and segno-mark-event (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

Clef_engraver (page 462)

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 to a clef. Values of 7 and -7 are common.

clefTranspositionStyle (symbol)

Determines the way the ClefModifier grob of a clef 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 565), and ClefModifier (page 568).

Collision_engraver (page 463)

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 656).

Cue_clef_engraver (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

clefTransposition (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Custos_engraver` (page 466)

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 583).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

`Horizontal_script_engraver` (page 477)

Aligns Script horizontally

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.

`Instrument_name_engraver` (page 477)

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 618).

Key_engraver (page 479)

Engrave a key signature.

Music types accepted: key-change-event (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

`Piano_pedal_engraver` (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

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 672), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `UnaCordaPedal` (page 739).

`Pure_from_neighbor_engraver` (page 494)

Coordinates items that get their pure heights from their neighbors.

`Rest_collision_engraver` (page 495)

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 679).

`Script_row_engraver` (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): `ScriptRow` (page 681).

`Separating_line_group_engraver` (page 496)

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 700).

`Signum_repetitionis_engraver` (page 497)

Create a `SignumRepetitionis` at the end of a `\repeat volta` section.

Music types accepted: `volta-repeat-end-event` (page 60),

This engraver creates the following layout object(s): `SignumRepetitionis` (page 685).

`Skip_typesetting_engraver` (page 497)

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 696).

`Staff_collecting_engraver` (page 499)

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 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Time_signature_engraver` (page 504)

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

2.1.19 `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 309).

This context creates the following layout object(s): `Accidental` (page 526),

`AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528),

`AccidentalSuggestion` (page 529), `BarLine` (page 538), `BassFigure` (page 544),

`BassFigureAlignment` (page 544), `BassFigureAlignmentPositioning` (page 545),

`BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), `BassFigureLine`

(page 548), `BreathingSign` (page 557), `CaesuraScript` (page 559), `Clef` (page 565),

`ClefModifier` (page 568), `CueClef` (page 577), `CueEndClef` (page 580), `DotColumn`

(page 588), `FingeringColumn` (page 605), `InstrumentName` (page 618), `KeyCancellation`

(page 622), `KeySignature` (page 625), `LedgerLineSpanner` (page 631), `NoteCollision`

(page 656), `OptionalMaterialBracket` (page 661), `OttavaBracket` (page 664),

`PianoPedalBracket` (page 672), `RestCollision` (page 679), `ScriptColumn` (page 680),

`ScriptRow` (page 681), `SostenutoPedal` (page 690), `SostenutoPedalLineSpanner` (page 692),

`StaffEllipsis` (page 696), `StaffHighlight` (page 699), `StaffSpacing` (page 700),

`StaffSymbol` (page 701), `SustainPedal` (page 710), `SustainPedalLineSpanner` (page 711),

`UnaCordaPedal` (page 739), `UnaCordaPedalLineSpanner` (page 740), and `VerticalAxisGroup` (page 742).

This context sets the following properties:

- Set context property `alterationGlyphs` to:

```
'((-1/2 . "accidentals.kievanM1")
```

```
(1/2 . "accidentals.kievan1"))
```

- Set context property `autoAccidentals` to:

```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```

- Set context property `autoCautionaries` to '().

- Set context property `caesuraType` to:

```
'((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 `positions` in `OptionalMaterialBracket` (page 661), to :

```
'(-1.75 . 1.75)
```
- Set grob property `protrusion` in `OptionalMaterialBracket` (page 661), to 0.5.
- Set grob property `thick-thickness` in `BarLine` (page 538), to 3.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `KievanVoice` (page 207).

Context `KievanStaff` can contain `CueVoice` (page 100), `KievanVoice` (page 207), and `NullVoice` (page 247).

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

`Accidental_engraver` (page 450)

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) . (alterbarnumber . measureposition)) pairs.

This engraver creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), and AccidentalSuggestion (page 529).

Alteration_glyph_engraver (page 451)

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 453)

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 742).

Bar_engraver (page 453)

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 49), caesura-event (page 51), coda-mark-event (page 51), dal-segno-event (page 51), fine-event (page 52), section-event (page 57), and segno-mark-event (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats (boolean)`

Notate volta-style repeats even when the repeat count is 1.

`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 return-count`

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

Caesura_engraver (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

Clef_engraver (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),
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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):
`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

Font_size_engraver (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

Grob_pq_engraver (page 476)

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.).

Horizontal_script_engraver (page 477)

Aligns Script horizontally

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.

Instrument_name_engraver (page 477)

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 618).

Key_engraver (page 479)

Engrave a key signature.

Music types accepted: `key-change-event` (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

Piano_pedal_engraver (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: *sostenuto*-event (page 57), *sustain*-event (page 59), and *una-corda*-event (page 60),

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 672), SostenutoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 495)

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 679).

Script_row_engraver (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Separating_line_group_engraver (page 496)

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 700).

Skip_typesetting_engraver (page 497)

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 696).

Staff_collecting_engraver (page 499)

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 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff_symbol_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

2.1.20 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 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), FingerGlideSpanner (page 601), Fingering (page 603), Flag (page 605), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), KievanLigature (page 628), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650),

MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), and VoiceFollower (page 744).

This context sets the following properties:

- Set context property `autoBeaming` to `#f`.
- Set grob property `duration-log` in `NoteHead` (page 658), to `note-head::calc-kievan-duration-log`.
- Set grob property `length` in `Stem` (page 703), to `0.0`.
- Set grob property `positions` in `Beam` (page 549), to `beam::get-kievan-positions`.
- Set grob property `quantized-positions` in `Beam` (page 549), to `beam::get-kievan-quantized-positions`.
- Set grob property `stencil` in `Flag` (page 605), to `#f`.
- Set grob property `stencil` in `Slur` (page 688), to `#f`.
- Set grob property `stencil` in `Stem` (page 703), to `#f`.
- Set grob property `style` in `Dots` (page 588), to `'kievan`.
- Set grob property `style` in `NoteHead` (page 658), to `'kievan`.
- Set grob property `style` in `Rest` (page 678), to `'mensural`.
- Set grob property `X-offset` in `Stem` (page 703), to `stem::kievan-offset-callback`.

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 452)

Generate an Arpeggio symbol.

Music types accepted: `arpeggio-event` (page 49),

This engraver creates the following layout object(s): `Arpeggio` (page 535).

`Auto_beam_engraver` (page 452)

Generate beams based on measure characteristics and observed Stems. Uses `beatBase`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through `Stem_engraver` (page 500), properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted: `beam-forbid-event` (page 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

`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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Beam_engraver` (page 457)

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

Music types accepted: `beam-event` (page 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Bend_engraver` (page 459)

Create fall spanners.

Music types accepted: `bend-after-event` (page 50),

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 551).

Breathing_sign_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

Chord_tremolo_engraver (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster_spanner_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double_percent_repeat_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Dynamic_align_engraver (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49), `break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two `Fingering`, `StringNumber` or `StrokeFinger` grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Fingering_engraver` (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

`Font_size_engraver` (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

Properties (read)

`glissandoMap` (list)

A map in the form of `'((source1 . target1) (source2 . target2) (source . 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 610).

`Grace_auto_beam_engraver` (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

`Grace_beam_engraver` (page 475)

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 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

Grace_engraver (page 475)

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 476)

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 478)

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 619).

Kievan_ligature_engraver (page 481)

Handle Kievan_ligature_events by glueing Kievan heads together.

Music types accepted: ligature-event (page 53),

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

Laissez_vibrer_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Multi_measure_rest_engraver (page 487)

Engrave multi-measure rests that are produced with 'R'. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54),

multi-measure-rest-event (page 54), and multi-measure-text-event (page 54),

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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

`New_fingering_engraver` (page 487)

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 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

`Note_head_line_engraver` (page 488)

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 744).

`Note_heads_engraver` (page 488)

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

Repeat_tie_engraver (page 494)

Create repeat ties.

Music types accepted: repeat-tie-event (page 56),

This engraver creates the following layout object(s): RepeatTie (page 676), and RepeatTieColumn (page 677).

Rest_engraver (page 495)

Engrave rests.

Music types accepted: rest-event (page 56),

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 678).

Rhythmic_column_engraver (page 495)

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

Script_column_engraver (page 495)

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 680).

Script_engraver (page 496)

Handle note scripted articulations.

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

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 679).

Slash_repeat_engraver (page 497)

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

Slur_engraver (page 497)

Build slur grobs from slur events.

Music types accepted: note-event (page 55), and slur-event (page 57),

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 688).

`Spanner_break_forbid_engraver` (page 499)

Forbid breaks in certain spanners.

`Stem_engraver` (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and TieColumn (page 727).

Trill_spanner_engraver (page 506)

Create trill spanners.

Music types accepted: trill-span-event (page 60),

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 734).

Tuplet_engraver (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 60),

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 736), and TupletNumber (page 737).

2.1.21 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 618), LyricExtender (page 635), LyricHyphen (page 636), LyricSpace (page 639), LyricText (page 640), StanzaNumber (page 702), VerticalAxisGroup (page 742), and VowelTransition (page 748).

This context sets the following properties:

- Set context property instrumentName to '().
- Set context property lyricRepeatCountFormatter to #<procedure at lily/translation-functions.scm:214:4 (context repeat-count)>.
- Set context property searchForVoice to #f.
- Set context property shortInstrumentName to '().
- Set grob property bar-extent in BarLine (page 538), to :
'(-0.05 . 0.05)
- Set grob property font-size in InstrumentName (page 618), to 1.0.
- Set grob property nonstaff-nonstaff-spacing in VerticalAxisGroup (page 742), to :
'((basic-distance . 0)
(minimum-distance . 2.8)
(padding . 0.2)
(stretchability . 0))

- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 742), to :
`'((basic-distance . 5.5)`
`(padding . 0.5)`
`(stretchability . 1))`
- Set grob property `nonstaff-unrelatedstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to 1.5.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 742), to #t.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 742), to #t.
- Set grob property `self-alignment-Y` in `InstrumentName` (page 618), to #f.
- Set grob property `short-bar-extent` in `BarLine` (page 538), to :
`'(-0.05 . 0.05)`
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), 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 453)

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 742).

`Extender_engraver` (page 471)

Create lyric extenders.

Music types accepted: `completize-extender-event` (page 51), and `extender-event` (page 52),

Properties (read)

`extendersOverRests` (boolean)

Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): `LyricExtender` (page 635).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

Hyphen_engraver (page 477)

Create lyric hyphens, vowel transitions and distance constraints between words.

Music types accepted: hyphen-event (page 53), and vowel-transition-event (page 60),

This engraver creates the following layout object(s): LyricHyphen (page 636), LyricSpace (page 639), and VowelTransition (page 748).

Instrument_name_engraver (page 477)

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 618).

Lyric_engraver (page 481)

Engrave text for lyrics.

Music types accepted: lyric-event (page 53),

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 640).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Stanza_number_engraver (page 500)

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 702).

2.1.22 MensuralStaff

Same as Staff context, except that it is accommodated for typesetting a piece in mensural style.

This context also accepts commands for the following context(s): Staff (page 309).

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), BreathingSign (page 557), CaesuraScript (page 559), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), Custos (page 583), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SignumRepetitionis (page 685), SostenuotoPedal (page 690), SostenuotoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), TimeSignature (page 727), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

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 at lily/music-functions.scm:1714:0 (context pitch barnum)>)'
```
- Set context property autoCautionaries to '().
- Set context property caesuraType to:


```
'((bar-line . "|"))'
```
- 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 doubleRepeatSegnoBarType to "S".
- Set context property endRepeatBarType to '().
- Set context property endRepeatSegnoBarType to "S".
- Set context property extraNatural to #f.
- 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 `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 `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 `hair-thickness` in `BarLine` (page 538), to 0.6.
- Set grob property `neutral-direction` in `Custos` (page 583), to -1.
- Set grob property `neutral-position` in `Custos` (page 583), to 3.
- Set grob property `style` in `Custos` (page 583), to 'mensural.
- Set grob property `style` in `TimeSignature` (page 727), to 'mensural.
- Set grob property `thick-thickness` in `BarLine` (page 538), to 1.8.
- Set grob property `thickness` in `StaffSymbol` (page 701), 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 235).

Context `MensuralStaff` can contain `CueVoice` (page 100), `MensuralVoice` (page 235), and `NullVoice` (page 247).

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

`Accidental_engraver` (page 450)

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 526), *AccidentalCautionary* (page 527), *AccidentalPlacement* (page 528), and *AccidentalSuggestion* (page 529).

Alteration_glyph_engraver (page 451)

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 453)

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 742).

Bar_engraver (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)
Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)
Notate volta-style repeats even when the repeat count is 1.

`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 return-count`
End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Custos_engraver` (page 466)

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 583).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

Font_size_engraver (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

Grob_pq_engraver (page 476)

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.).

Horizontal_script_engraver (page 477)

Aligns Script horizontally

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.

Instrument_name_engraver (page 477)

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 618).

Key_engraver (page 479)

Engrave a key signature.

Music types accepted: `key-change-event` (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

Piano_pedal_engraver (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: *sostenuto*-event (page 57), *sustain*-event (page 59), and *una-corda*-event (page 60),

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 672), SostenutoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure_from_neighbor_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest_collision_engraver (page 495)

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 679).

Script_row_engraver (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Separating_line_group_engraver (page 496)

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 700).

Signum_repetitionis_engraver (page 497)

Create a SignumRepetitionis at the end of a \repeat volta section.

Music types accepted: volta-repeat-end-event (page 60),

This engraver creates the following layout object(s): SignumRepetitionis (page 685).

Skip_typesetting_engraver (page 497)

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 696).

Staff_collecting_engraver (page 499)

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 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff_symbol_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

Time_signature_engraver (page 504)

Create a TimeSignature (page 727), whenever timeSignatureFraction changes.

Music types accepted: time-signature-event (page 59),

Properties (read)

initialTimeSignatureVisibility (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

2.1.23 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 439).

This context creates the following layout object(s): `Arpeggio` (page 535), `Beam` (page 549), `BendAfter` (page 551), `BreathingSign` (page 557), `ClusterSpanner` (page 569), `ClusterSpannerBeacon` (page 570), `CombineTextScript` (page 572), `Dots` (page 588), `DoublePercentRepeat` (page 590), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `FingerGlideSpanner` (page 601), `Fingering` (page 603), `Flag` (page 605), `Glissando` (page 610), `Hairpin` (page 614), `InstrumentSwitch` (page 619), `LaissezVibrerTie` (page 629), `LaissezVibrerTieColumn` (page 630), `MensuralLigature` (page 646), `MultiMeasureRest` (page 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), `MultiMeasureRestText` (page 653), `NoteColumn` (page 657), `NoteHead` (page 658), `NoteSpacing` (page 660), `PercentRepeat` (page 667), `PercentRepeatCounter` (page 668), `PhrasingSlur` (page 670), `RepeatSlash` (page 675), `RepeatTie` (page 676), `RepeatTieColumn` (page 677), `Rest` (page 678), `Script` (page 679), `ScriptColumn` (page 680), `Stem` (page 703), `StemStub` (page 705), `StemTremolo` (page 705), `StringNumber` (page 706), `StrokeFinger` (page 708), `TextScript` (page 721), `TextSpanner` (page 723), `Tie` (page 725), `TieColumn` (page 727), `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), `TrillPitchParentheses` (page 733), `TrillSpanner` (page 734), `TupletBracket` (page 736), `TupletNumber` (page 737), and `VoiceFollower` (page 744).

This context sets the following properties:

- Set context property `autoBeaming` to `#f`.
- Set grob property `style` in `Flag` (page 605), to `'mensural'`.
- Set grob property `style` in `NoteHead` (page 658), to `'mensural'`.
- Set grob property `style` in `Rest` (page 678), 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 452)

Generate an `Arpeggio` symbol.

Music types accepted: `arpeggio-event` (page 49),

This engraver creates the following layout object(s): `Arpeggio` (page 535).

`Auto_beam_engraver` (page 452)

Generate beams based on measure characteristics and observed Stems. Uses `beatBase`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through `Stem_engraver` (page 500), properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Bend_engraver (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

`Breathing_sign_engraver` (page 460)

Notate breath marks.

Music types accepted: `breathing-event` (page 51),

Properties (read)

`breathMarkType` (symbol)

The type of `BreathingSign` to create at `\breathe`.

This engraver creates the following layout object(s): `BreathingSign` (page 557).

`Chord_tremolo_engraver` (page 462)

Generate beams for tremolo repeats.

Music types accepted: `tremolo-span-event` (page 59),

This engraver creates the following layout object(s): `Beam` (page 549).

`Cluster_spanner_engraver` (page 463)

Engrave a cluster using `Spanner` notation.

Music types accepted: `cluster-note-event` (page 51),

This engraver creates the following layout object(s): `ClusterSpanner` (page 569), and `ClusterSpannerBeacon` (page 570).

`Dots_engraver` (page 468)

Create `Dots` (page 588), objects for `rhythmic-head-interface` (page 813)s.

This engraver creates the following layout object(s): `Dots` (page 588).

`Double_percent_repeat_engraver` (page 468)

Make double measure repeats.

Music types accepted: `double-percent-event` (page 51),

Properties (read)

`countPercentRepeats` (boolean)

If set, produce counters for percent repeats.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`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 590), and `DoublePercentRepeatCounter` (page 591).

`Dynamic_align_engraver` (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49), `break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two Fingering, StringNumber or StrokeFinger grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Fingering_engraver` (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

`Font_size_engraver` (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

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 610).

Grace_auto_beam_engraver (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

Grace_beam_engraver (page 475)

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 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace_engraver (page 475)

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 476)

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 478)

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 619).

Laissez_vibrer_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Mensural_ligature_engraver (page 485)

Handle Mensural_ligature_events by glueing special ligature heads together.

Music types accepted: ligature-event (page 53),

This engraver creates the following layout object(s): MensuralLigature (page 646).

Multi_measure_rest_engraver (page 487)

Engrave multi-measure rests that are produced with 'R'. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54), multi-measure-rest-event (page 54), and multi-measure-text-event (page 54), 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 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), and MultiMeasureRestText (page 653).

New_fingering_engraver (page 487)

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 603), Script (page 679), StringNumber (page 706), and StrokeFinger (page 708).

Note_head_line_engraver (page 488)

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 744).

Note_heads_engraver (page 488)

Generate note heads.

Music types accepted: note-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.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

Phrasing_slur_engraver (page 492)

Print phrasing slurs. Similar to **Slur_engraver** (page 497).

Music types accepted: **note-event** (page 55), and **phrasing-slur-event** (page 56),

This engraver creates the following layout object(s): **PhrasingSlur** (page 670).

Pitched_trill_engraver (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): **TrillPitchAccidental** (page 730), **TrillPitchGroup** (page 731), **TrillPitchHead** (page 733), and **TrillPitchParentheses** (page 733).

Repeat_tie_engraver (page 494)

Create repeat ties.

Music types accepted: **repeat-tie-event** (page 56),

This engraver creates the following layout object(s): **RepeatTie** (page 676), and **RepeatTieColumn** (page 677).

Rest_engraver (page 495)

Engrave rests.

Music types accepted: **rest-event** (page 56),

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 678).

Rhythmic_column_engraver (page 495)

Generate **NoteColumn**, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): **NoteColumn** (page 657).

Script_column_engraver (page 495)

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 680).

Script_engraver (page 496)

Handle note scripted articulations.

Music types accepted: **articulation-event** (page 50),

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 679).

Slash_repeat_engraver (page 497)

Make beat repeats.

Music types accepted: **repeat-slash-event** (page 56),

This engraver creates the following layout object(s): **DoubleRepeatSlash** (page 592), and **RepeatSlash** (page 675).

`Spanner_break_forbid_engraver` (page 499)

Forbid breaks in certain spanners.

`Stem_engraver` (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and TieColumn (page 727).

Trill_spanner_engraver (page 506)

Create trill spanners.

Music types accepted: trill-span-event (page 60),

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 734).

Tuplet_engraver (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: tuplet-span-event (page 60),

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 736), and TupletNumber (page 737).

2.1.24 NoteNames

Typesets note names.

This context also accepts commands for the following context(s): Staff (page 309).

This context creates the following layout object(s): NoteName (page 659), StaffSpacing (page 700), Tie (page 725), TieColumn (page 727), and VerticalAxisGroup (page 742).

This context sets the following properties:

- Set grob property nonstaff-nonstaff-spacing in VerticalAxisGroup (page 742), to :
'((basic-distance . 0)
(minimum-distance . 2.8)
(padding . 0.2)
(stretchability . 0))
- Set grob property nonstaff-relatedstaff-spacing in VerticalAxisGroup (page 742), to :
'((basic-distance . 5.5)
(padding . 0.5)
(stretchability . 1))
- Set grob property nonstaff-unrelatedstaff-spacing.padding in VerticalAxisGroup (page 742), to 1.5.
- Set grob property staff-affinity in VerticalAxisGroup (page 742), 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 451)

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 453)

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 742).

`Note_name_engraver` (page 488)

Print pitches as words.

Music types accepted: `note-event` (page 55),

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 659).

Separating_line_group_engraver (page 496)

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 700).

Tie_engraver (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 59),

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 725), and TieColumn (page 727).

2.1.25 NullVoice

For aligning lyrics without printing notes.

This context also accepts commands for the following context(s): Staff (page 309), and Voice (page 439).

This context creates the following layout object(s): Beam (page 549), NoteHead (page 658), Slur (page 688), Tie (page 725), and TieColumn (page 727).

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 658), to #t.
- Set grob property stencil in Beam (page 549), to #f.
- Set grob property stencil in NoteHead (page 658), to #f.
- Set grob property stencil in Slur (page 688), to #f.
- Set grob property stencil in Tie (page 725), to #f.
- Set grob property X-extent in NoteHead (page 658), to #<procedure 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 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grob_pq_engraver (page 476)

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.).

Note_heads_engraver (page 488)

Generate note heads.

Music types accepted: note-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.

staffLineLayoutFunction (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): NoteHead (page 658).

Pitch_squash_engraver (page 493)

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 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and `TieColumn` (page 727).

2.1.26 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 742).

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `Staff` (page 309).

Context `OneStaff` can contain `ChordNames` (page 98), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

Axis_group_engraver (page 453)

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 742).

2.1.27 PetrucciStaff

A kind of Staff approximating the mensural typesetting of Ottaviano Petrucci's *Harmonices Musices Odhecaton* (Venice, 1501).

This context also accepts commands for the following context(s): Staff (page 309).

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), BreathingSign (page 557), CaesuraScript (page 559), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), Custos (page 583), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SignumRepetitionis (page 685), SostenutoPedal (page 690), SostenutoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), TimeSignature (page 727), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

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 at lily/music-functions.scm:1714:0 (context pitch barnum)>
 #<procedure neo-modern-accidental-rule (context pitch barnum)>)
```

- Set context property `autoAccidentals` to:
`'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)`
- Set context property `autoCautionaries` to `'()`.
- Set context property `caesuraType` to:
`'((bar-line . "|"))`
- Set context property `clefGlyph` to `"clefs.petrucchi.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 `doubleRepeatSegnoBarType` to `"S"`.
- Set context property `endRepeatBarType` to `'()`.
- Set context property `endRepeatSegnoBarType` to `"S"`.
- Set context property `extraNatural` to `#f`.
- 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 `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 `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 `underlyingRepeatBarType` to `'()`.
- Set grob property `bar-extent` in `BarLine` (page 538), to :
`'(-2.5 . 2.5)`
- Set grob property `bar-extent` in `SignumRepetitionis` (page 685), to :
`'(-2.5 . 2.5)`
- Set grob property `hair-thickness` in `BarLine` (page 538), to `2.21`.

- Set grob property hair-thickness in SignumRepetitionis (page 685), to 2.21.
- Set grob property kern in BarLine (page 538), to 2.9.
- Set grob property kern in SignumRepetitionis (page 685), to 2.9.
- Set grob property neutral-direction in Custos (page 583), to -1.
- Set grob property neutral-position in Custos (page 583), to 3.
- Set grob property rounded in BarLine (page 538), to #t.
- Set grob property rounded in SignumRepetitionis (page 685), to #t.
- Set grob property short-bar-extent in BarLine (page 538), to :
'(-1.5 . 1.5)
- Set grob property short-bar-extent in SignumRepetitionis (page 685), to :
'(-1.5 . 1.5)
- Set grob property style in Custos (page 583), to 'mensural.
- Set grob property style in TimeSignature (page 727), to 'mensural.
- Set grob property thick-thickness in BarLine (page 538), to 2.9.
- Set grob property thick-thickness in SignumRepetitionis (page 685), to 2.9.
- Set grob property thickness in StaffSymbol (page 701), 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 265).

Context PetrucciStaff can contain CueVoice (page 100), NullVoice (page 247), and PetrucciVoice (page 265).

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

Accidental_engraver (page 450)

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 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), and `AccidentalSuggestion` (page 529).

`Alteration_glyph_engraver` (page 451)

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 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)
 Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and CaesuraScript (page 559).

Clef_engraver (page 462)

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 to a clef. Values of 7 and -7 are common.

clefTranspositionStyle (symbol)

Determines the way the ClefModifier grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Custos_engraver` (page 466)

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 583).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544),

`BassFigureAlignment` (page 544), `BassFigureBracket` (page 546),

`BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

Horizontal_script_engraver (page 477)

Aligns Script horizontally

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.

Instrument_name_engraver (page 477)

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 618).

Key_engraver (page 479)

Engrave a key signature.

Music types accepted: *key-change-event* (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

`Piano_pedal_engraver` (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

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 672), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `UnaCordaPedal` (page 739).

`Pure_from_neighbor_engraver` (page 494)

Coordinates items that get their pure heights from their neighbors.

`Rest_collision_engraver` (page 495)

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 679).

`Script_row_engraver` (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): `ScriptRow` (page 681).

`Separating_line_group_engraver` (page 496)

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 700).

`Signum_repetitionis_engraver` (page 497)

Create a `SignumRepetitionis` at the end of a `\repeat volta` section.

Music types accepted: `volta-repeat-end-event` (page 60),

This engraver creates the following layout object(s): `SignumRepetitionis` (page 685).

`Skip_typesetting_engraver` (page 497)

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 696).

`Staff_collecting_engraver` (page 499)

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 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Time_signature_engraver` (page 504)

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

2.1.28 PetrucciVoice

A kind of Voice approximating the mensural typesetting of Ottaviano Petrucci's *Harmonices Musices Odhecaton* (Venice, 1501).

This context also accepts commands for the following context(s): Voice (page 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), FingerGlideSpanner (page 601), Fingering (page 603), Flag (page 605), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), MensuralLigature (page 646), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), and VoiceFollower (page 744).

This context sets the following properties:

- Set context property `autoBeaming` to `#f`.
- Set grob property `length` in Stem (page 703), to 5.
- Set grob property `style` in Flag (page 605), to 'mensural'.
- Set grob property `style` in NoteHead (page 658), to 'petrucci'.
- Set grob property `style` in Rest (page 678), to 'mensural'.
- Set grob property `thickness` in Stem (page 703), 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 452)

Generate an Arpeggio symbol.

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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto_beam_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses `beatBase`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through Stem_engraver (page 500), properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

`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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Beam_engraver` (page 457)

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

Music types accepted: `beam-event` (page 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Bend_engraver` (page 459)

Create fall spanners.

Music types accepted: `bend-after-event` (page 50),

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 551).

`Breathing_sign_engraver` (page 460)

Notate breath marks.

Music types accepted: `breathing-event` (page 51),

Properties (read)

`breathMarkType` (symbol)

The type of `BreathingSign` to create at `\breathe`.

This engraver creates the following layout object(s): `BreathingSign` (page 557).

`Chord_tremolo_engraver` (page 462)

Generate beams for tremolo repeats.

Music types accepted: `tremolo-span-event` (page 59),

This engraver creates the following layout object(s): `Beam` (page 549).

`Cluster_spanner_engraver` (page 463)

Engrave a cluster using `Spanner` notation.

Music types accepted: `cluster-note-event` (page 51),

This engraver creates the following layout object(s): `ClusterSpanner` (page 569), and `ClusterSpannerBeacon` (page 570).

`Dots_engraver` (page 468)

Create `Dots` (page 588), objects for `rhythmic-head-interface` (page 813)s.

This engraver creates the following layout object(s): `Dots` (page 588).

`Double_percent_repeat_engraver` (page 468)

Make double measure repeats.

Music types accepted: `double-percent-event` (page 51),

Properties (read)

`countPercentRepeats` (boolean)

If set, produce counters for percent repeats.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`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 590), and `DoublePercentRepeatCounter` (page 591).

`Dynamic_align_engraver` (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49), `break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two `Fingering`, `StringNumber` or `StrokeFinger` grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Fingering_engraver` (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

`Font_size_engraver` (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: glissando-event (page 52),

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 610).

Grace_auto_beam_engraver (page 474)

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 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): Beam (page 549).

Grace_beam_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace_engraver (page 475)

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 476)

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 478)

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 619).

Laissez_vibrer_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Mensural_ligature_engraver (page 485)

Handle Mensural_ligature_events by glueing special ligature heads together.

Music types accepted: ligature-event (page 53),

This engraver creates the following layout object(s): MensuralLigature (page 646).

Multi_measure_rest_engraver (page 487)

Engrave multi-measure rests that are produced with 'R'. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54), multi-measure-rest-event (page 54), and multi-measure-text-event (page 54), 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 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), and MultiMeasureRestText (page 653).

New_fingering_engraver (page 487)

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 603), Script (page 679), StringNumber (page 706), and StrokeFinger (page 708).

Note_head_line_engraver (page 488)

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 744).

Note_heads_engraver (page 488)

Generate note heads.

Music types accepted: note-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.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

Phrasing_slur_engraver (page 492)

Print phrasing slurs. Similar to Slur_engraver (page 497).

Music types accepted: note-event (page 55), and phrasing-slur-event (page 56),

This engraver creates the following layout object(s): PhrasingSlur (page 670).

Pitched_trill_engraver (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), and TrillPitchParentheses (page 733).

Repeat_tie_engraver (page 494)

Create repeat ties.

Music types accepted: repeat-tie-event (page 56),

This engraver creates the following layout object(s): RepeatTie (page 676), and RepeatTieColumn (page 677).

Rest_engraver (page 495)

Engrave rests.

Music types accepted: rest-event (page 56),

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 678).

Rhythmic_column_engraver (page 495)

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

Script_column_engraver (page 495)

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 680).

Script_engraver (page 496)

Handle note scripted articulations.

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

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 679).

Slash_repeat_engraver (page 497)

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

Slur_engraver (page 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

Spanner_break_forbid_engraver (page 499)

Forbid breaks in certain spanners.

Stem_engraver (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

Text_engraver (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

Text_spanner_engraver (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

Tie_engraver (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and `TieColumn` (page 727).

`Trill_spanner_engraver` (page 506)

Create trill spanners.

Music types accepted: `trill-span-event` (page 60),

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 734).

`Tuplet_engraver` (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

2.1.29 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 139).

This context creates the following layout object(s): `Arpeggio` (page 535), `InstrumentName` (page 618), `SpanBar` (page 694), `SpanBarStub` (page 695), `StaffGrouper` (page 698), `SystemStartBar` (page 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), `SystemStartSquare` (page 716), and `VerticalAlignment` (page 742).

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 597), to `#f`.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `Staff` (page 309).

Context `PianoStaff` can contain `ChoirStaff` (page 67), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Instrument_name_engraver` (page 477)

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 618).

`Keep_alive_together_engraver` (page 479)

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 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Span_arpeggio_engraver` (page 498)

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 535).

`Span_bar_engraver` (page 498)

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 694).

`Span_bar_stub_engraver` (page 498)

Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s): `SpanBarStub` (page 695).

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Vertical_align_engraver` (page 507)

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): `StaffGroup` (page 698), and `VerticalAlignment` (page 742).

2.1.30 RhythmicStaff

Like `Staff` but for printing rhythms. Pitches are ignored when engraving; the notes are printed on one line. The MIDI rendition retains pitches unchanged.

This context also accepts commands for the following context(s): `Staff` (page 309).

This context creates the following layout object(s): BarLine (page 538), BreathingSign (page 557), CaesuraScript (page 559), DotColumn (page 588), InstrumentName (page 618), LedgerLineSpanner (page 631), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), TimeSignature (page 727), and VerticalAxisGroup (page 742).

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 701), to 1.
- Set grob property neutral-direction in Beam (page 549), to 1.
- Set grob property neutral-direction in Stem (page 703), to 1.
- Set grob property staff-padding in VoltaBracket (page 745), to 3.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type Voice (page 439).

Context RhythmicStaff can contain CueVoice (page 100), NullVoice (page 247), and Voice (page 439).

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

Axis_group_engraver (page 453)

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 742).

Bar_engraver (page 453)

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 49), caesura-event (page 51), coda-mark-event (page 51), dal-segno-event (page 51), fine-event (page 52), section-event (page 57), and segno-mark-event (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

`Dot_column_engraver` (page 467)

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 588).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Instrument_name_engraver` (page 477)

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 618).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Pitch_squash_engraver` (page 493)

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 496)

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 700).

`Staff_highlight_engraver` (page 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Time_signature_engraver` (page 504)

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

2.1.31 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 called `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 283).

This context creates the following layout object(s): `BarNumber` (page 542), `BreakAlignGroup` (page 554), `BreakAlignment` (page 555), `CenteredBarNumber` (page 561), `CenteredBarNumberLineSpanner` (page 562), `CodaMark` (page 570), `ControlPoint` (page 575), `ControlPolygon` (page 576), `Footnote` (page 606), `GraceSpacing` (page 611), `JumpScript`

(page 621), LeftEdge (page 632), MetronomeMark (page 647), NonMusicalPaperColumn (page 655), PaperColumn (page 665), Parentheses (page 666), RehearsalMark (page 673), SectionLabel (page 681), SegnoMark (page 683), SpacingSpanner (page 693), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TextMark (page 719), VerticalAlignment (page 742), VoltaBracket (page 745), and VoltaBracketSpanner (page 747).

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
  measureStartNow
  lastChord)
```
- Set context property associatedVoiceType to 'Voice.
- Set context property autoAccidentals to:


```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```
- 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-
- Set context property beamHalfMeasure to #t.
- Set context property breathMarkDefinitions to:


```
'((altcomma
  (text #<procedure musicglyph-markup (layout props glyph-name)>
    "scripts.raltcomma"))
  (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
(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.uupbow"))
(varcomma
(text #<procedure musicglyph-markup (layout props glyph-name)>
  "scripts.rvarcomma"))

```

- 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 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 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 at lily/translation-functions.scm:228:4 (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>`.
- 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)
 (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 "m".
  - Set context property noChordSymbol to "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 printTrivialVoltaRepeats to #f.
  - 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 at lily/translation-functions.scm:228:4 (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-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
 (altcomma
 (script-stencil feta "laltcomma" . "raltcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
 (circulus
 (script-stencil feta "circulus" . "circulus")
 (side-relative-direction . -1)
 (avoid-slur . ignore)
 (padding . 0.2)
 (quantize-position . #t)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
 (coda (script-stencil feta "coda" . "coda")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
 (comma (script-stencil feta "lcomma" . "rcomma")
 (quantize-position . #t)
 (padding . 0.2)

```

```

 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(downbow
 (script-stencil feta "ddownbow" . "udownbow")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(downmordent
 (script-stencil
 feta
 "downmordent"
 .
 "downmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(downprall
 (script-stencil feta "downprall" . "downprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(espressivo
 (avoid-slur . around)
 (padding . 0.2)
 (script-stencil feta "espr" . "espr")
 (side-axis . 1)
 (side-relative-direction . -1))
(fermata
 (script-stencil feta "dfermata" . "ufermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(flageolet
 (script-stencil feta "flageolet" . "flageolet")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 50))
(halfopen
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "halfopen" . "halfopen"))

```

```

 (side-axis . 1)
 (direction . 1))
(halfopenvertical
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil
 feta
 "halfopenvertical"
 .
 "halfopenvertical")
 (side-axis . 1)
 (direction . 1))
(haydnturn
 (script-stencil feta "haydnturn" . "haydnturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))
(henzelongfermata
 (script-stencil
 feta
 "dhenzelongfermata"
 .
 "uhenzelongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(henzeshortfermata
 (script-stencil
 feta
 "dhenzeshortfermata"
 .
 "uhenzeshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ictus (script-stencil feta "ictus" . "ictus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (side-axis . 1)
 (direction . -1))
(lheel (script-stencil feta "upedalheel" . "upedalheel")
 (padding . 0.2)

```

```

 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(lineprall
 (script-stencil feta "lineprall" . "lineprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(longfermata
 (script-stencil
 feta
 "dlongfermata"
 .
 "ulongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ltoe (script-stencil feta "upedaltoe" . "upedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(marcato
 (script-stencil feta "dmarcato" . "umarcato")
 (padding . 0.2)
 (avoid-slur . inside)
 (quantize-position . #t)
 (side-axis . 1)
 (side-relative-direction . -1))
(mordent
 (script-stencil feta "mordent" . "mordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(open (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "open" . "open")
 (side-axis . 1)
 (direction . 1))
(outsidecomma
 (avoid-slur . around)
 (direction . 1)
 (padding . 0.2)
 (side-axis . 1)
 (script-stencil feta "lcomma" . "rcomma"))
(portato
 (script-stencil feta "uportato" . "dportato"))

```

```

 (avoid-slur . around)
 (padding . 0.45)
 (side-axis . 1)
 (side-relative-direction . -1))
(prall (script-stencil feta "prall" . "prall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(pralldown
 (script-stencil feta "pralldown" . "pralldown")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallmordent
 (script-stencil
 feta
 "prallmordent"
 .
 "prallmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallprall
 (script-stencil feta "prallprall" . "prallprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallup
 (script-stencil feta "prallup" . "prallup")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(reverseturn
 (script-stencil
 feta
 "reverseturn"
 .
 "reverseturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))
(rheel (script-stencil feta "dpedalheel" . "dpedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))

```

```

(rtoe (script-stencil feta "dpedaltoe" . "dpedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(segno (script-stencil feta "segno" . "segno")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(semicirculus
 (script-stencil
 feta
 "dsemicirculus"
 .
 "dsemicirculus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
(shortfermata
 (script-stencil
 feta
 "dshortfermata"
 .
 "ushortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(signumcongruentiae
 (script-stencil
 feta
 "dsignumcongruentiae"
 .
 "usignumcongruentiae")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(slashturn
 (script-stencil feta "slashturn" . "slashturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))
(snappizzicato

```



```

(script-stencil
 feta
 "snappizzicato"
 .
 "snappizzicato")
(padding . 0.2)
(avoid-slur . outside)
(side-axis . 1)
(direction . 1))
(staccatissimo
 (avoid-slur . inside)
 (quantize-position . #t)
 (script-stencil
 feta
 "dstaccatissimo"
 .
 "ustaccatissimo")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.1)
 (side-axis . 1)
 (side-relative-direction . -1)
 (toward-stem-shift . 1.0)
 (toward-stem-shift-in-column . 0.0))
(staccato
 (script-stencil feta "staccato" . "staccato")
 (side-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
(tenuto
 (script-stencil feta "tenuto" . "tenuto")
 (quantize-position . #t)
 (avoid-slur . inside)
 (padding . 0.2)
 (script-priority . -50)
 (side-axis . 1)
 (side-relative-direction . -1))
(trill (script-stencil feta "trill" . "trill")
 (direction . 1)
 (padding . 0.2)
 (avoid-slur . outside)

```

```

 (side-axis . 1)
 (script-priority . 150))
(turn (script-stencil feta "turn" . "turn")
 (avoid-slur . inside)
 (padding . 0.2)
 (side-axis . 1)
 (direction . 1))
(upbow (script-stencil feta "dupbow" . "uupbow")
 (avoid-slur . around)
 (padding . 0.2)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(upmordent
 (script-stencil feta "upmordent" . "upmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(upprall
 (script-stencil feta "upprall" . "upprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(varcoda
 (script-stencil feta "varcoda" . "varcoda")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(varcomma
 (script-stencil feta "lvarcomma" . "rvarcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(verylongfermata
 (script-stencil
 feta
 "dverylongfermata"
 .
 "uverylongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(veryshortfermata
 (script-stencil

```

```

feta
"dveryshortfermata"
.
"uveryshortfermata")
(padding . 0.4)
(avoid-slur . around)
(outside-staff-priority . 75)
(script-priority . 175)
(side-axis . 1)
(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 "/".
- Set context property soloIIIText 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 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 4))))
((9 . 4)
 (beamExceptions (end (1/32 8 8 8 8 8 8 8 8))))
((12 . 4)
 (beamExceptions
 (end (1/32 8 8 8 8 8 8 8 8 8 8 8 8))))
((5 . 8) (beatStructure 3 2))
((8 . 8) (beatStructure 3 3 2)))

```

- Set 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 309).

Context `Score` can contain `ChoirStaff` (page 67), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Bar_number_engraver` (page 456)

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 `Staff_collecting_engraver` (page 499). 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 first volta number for the current \alternative element.  
Not set outside of alternatives.

`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)

The current point within the measure.

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): `BarNumber` (page 542), and `CenteredBarNumber` (page 561).

Beam\_collision\_engraver (page 457)

Help beams avoid colliding with notes and clefs in other voices.

Break\_align\_engraver (page 460)

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 554), BreakAlignment (page 555), and LeftEdge (page 632).

Centered\_bar\_number\_align\_engraver (page 461)

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 562).

Concurrent\_hairpin\_engraver (page 465)

Collect concurrent hairpins.

Footnote\_engraver (page 473)

Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 606).

Grace\_spacing\_engraver (page 475)

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 611).

Jump\_engraver (page 478)

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 499), also needs to be there so that marks appear at the intended Y location.

Music types accepted: ad-hoc-jump-event (page 49), dal-segno-event (page 51), and fine-event (page 52),

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 621).

`Mark_engraver` (page 482)

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 499), 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 483). 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 570), `RehearsalMark` (page 673), `SectionLabel` (page 681), and `SegnoMark` (page 683).

`Mark_tracking_translator` (page 483)

This translator chooses which marks `Mark_engraver` should engrave.

Music types accepted: `ad-hoc-mark-event` (page 49), `coda-mark-event` (page 51), `rehearsal-mark-event` (page 56), `section-label-event` (page 57), and `segno-mark-event` (page 57),

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 485)

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 Staff\_collecting\_engraver (page 499).

Music types accepted: tempo-change-event (page 59),

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 647).

Output\_property\_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Paper\_column\_engraver (page 490)

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 50), and label-event (page 53),

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 655), and `PaperColumn` (page 665).

`Parenthesis_engraver` (page 491)

Parenthesize objects whose `parenthesize` property is `#t`.

This engraver creates the following layout object(s): `Parentheses` (page 666).

`Repeat_acknowledge_engraver` (page 494)

This translator adds entries to `repeatCommands` for events generated by `\\repeat volta`.

Music types accepted: `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 return-count`

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 496)

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 575), and `ControlPolygon` (page 576).

`Spacing_engraver` (page 498)

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted: `spacing-section-event` (page 57),

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` (non-negative exact rational or `+inf.0`)

Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s): `SpacingSpanner` (page 693).

`Spanner_tracking_engraver` (page 499)

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 499)

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 500)

This engraver ensures that stanza numbers are neatly aligned.

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Text_mark_engraver` (page 503)

Engraves arbitrary textual marks.

Music types accepted: `text-mark-event` (page 59),

Properties (read)

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): `TextMark` (page 719).

`Timing_translator` (page 505)

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 49), `bar-check-event` (page 50), `bar-event` (page 50), `fine-event` (page 52), and `partial-event` (page 55),

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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`timeSignatureFraction` (simple time signature)

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 first volta number for the current \alternative element. Not set outside of alternatives.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`timeSignatureFraction` (simple time signature)

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

`Tweak_engraver` (page 507)

Read the tweaks property from the originating event, and set properties.

`Vertical_align_engraver` (page 507)

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 698), and `VerticalAlignment` (page 742).

`Volta_engraver` (page 507)

Make volta brackets.

Music types accepted: `dal-segno-event` (page 51), `fine-event` (page 52), and `volta-span-event` (page 60),

Properties (read)

`currentCommandColumn` (graphical (layout) object)

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

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*  
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.  
`start-repeat` *repeat-count*  
 Start a repeated section. *repeat-count* is the number of times to perform this 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.

This engraver creates the following layout object(s): `VoltaBracket` (page 745), and `VoltaBracketSpanner` (page 747).

## 2.1.32 Staff

Handles clefs, bar lines, keys, accidentals. It can contain Voice contexts.

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), BreathingSign (page 557), CaesuraScript (page 559), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SostenutoPedal (page 690), SostenutoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), TimeSignature (page 727), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

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 439).

Context Staff can contain CueVoice (page 100), NullVoice (page 247), and Voice (page 439).

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

Accidental\_engraver (page 450)

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) . (alterbarnumber . measureposition)) pairs.

This engraver creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), and AccidentalSuggestion (page 529).

## Alteration\_glyph\_engraver (page 451)

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 453)

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 742).

## Bar\_engraver (page 453)

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 49), caesura-event (page 51), coda-mark-event (page 51), dal-segno-event (page 51), fine-event (page 52), section-event (page 57), and segno-mark-event (page 57),

## 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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats (boolean)`

Notate volta-style repeats even when the repeat count is 1.

`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 return-count`

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

Caesura\_engraver (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

Clef\_engraver (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

## Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),  
 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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):  
`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

Font\_size\_engraver (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

Grob\_pq\_engraver (page 476)

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.).

Horizontal\_script\_engraver (page 477)

Aligns Script horizontally

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.

Instrument\_name\_engraver (page 477)

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 618).

Key\_engraver (page 479)

Engrave a key signature.

Music types accepted: `key-change-event` (page 53),

## 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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).



Piano\_pedal\_engraver (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: *sostenuto*-event (page 57), *sustain*-event (page 59), and *una-corda*-event (page 60),

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 672), SostenutoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure\_from\_neighbor\_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest\_collision\_engraver (page 495)

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 679).

Script\_row\_engraver (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Separating\_line\_group\_engraver (page 496)

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 700).

Skip\_typesetting\_engraver (page 497)

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 696).

Staff\_collecting\_engraver (page 499)

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 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff\_symbol\_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

Time\_signature\_engraver (page 504)

Create a TimeSignature (page 727), whenever timeSignatureFraction changes.

Music types accepted: time-signature-event (page 59),

Properties (read)

initialTimeSignatureVisibility (vector)

break visibility for the initial time signature.

partialBusy (boolean)

Signal that \partial acts at the current timestep.

timeSignatureFraction (simple time signature)

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 727).

### 2.1.33 StaffGroup

Connect staves vertically by adding a bracket on the left side. The bar lines of the contained staves are connected vertically, too.

This context creates the following layout object(s): Arpeggio (page 535), InstrumentName (page 618), SpanBar (page 694), SpanBarStub (page 695), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), and VerticalAlignment (page 742).

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 597), to #f.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `Staff` (page 309).

Context `StaffGroup` can contain `ChoirStaff` (page 67), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Instrument_name_engraver` (page 477)

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 618).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Span_arpeggio_engraver` (page 498)

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 535).

`Span_bar_engraver` (page 498)

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 694).

`Span_bar_stub_engraver` (page 498)

Make stubs for span bars in all contexts that the span bars cross.

This engraver creates the following layout object(s): `SpanBarStub` (page 695).

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Vertical_align_engraver` (page 507)

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): `StaffGroup` (page 698), and `VerticalAlignment` (page 742).

### 2.1.34 `StandaloneRhythmScore`

A Score-level context for use by `\markup \rhythm`.

This context also accepts commands for the following context(s): `Score` (page 283), and `Timing` (page 283).

This context creates the following layout object(s): BarNumber (page 542), BreakAlignGroup (page 554), BreakAlignment (page 555), CenteredBarNumber (page 561), CenteredBarNumberLineSpanner (page 562), CodaMark (page 570), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), GraceSpacing (page 611), JumpScript (page 621), LeftEdge (page 632), MetronomeMark (page 647), NonMusicalPaperColumn (page 655), PaperColumn (page 665), Parentheses (page 666), RehearsalMark (page 673), SectionLabel (page 681), SegnoMark (page 683), SpacingSpanner (page 693), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TextMark (page 719), VerticalAlignment (page 742), VoltaBracket (page 745), and VoltaBracketSpanner (page 747).

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
 measureStartNow
 lastChord)
```
- Set context property associatedVoiceType to 'Voice.
- Set context property autoAccidentals to:
 

```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```
- 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-
- Set context property beamHalfMeasure to #t.
- Set context property breathMarkDefinitions to:
 

```
'((altcomma
 (text #<procedure musicglyph-markup (layout props glyph-name)>
 "scripts.raltcomma"))
 (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
 (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.uupbow"))
(varcomma
 (text #<procedure musicglyph-markup (layout props glyph-name)>
 "scripts.rvarcomma")))

```

- 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 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 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 at lily/translation-functions.scm:228:4 (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>.
- 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)
 (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 "m".
- Set context property noChordSymbol to "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 `printTrivialVoltaRepeats` to `#f`.
- 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 at lily/translation-functions.scm:228:4 (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-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
 (altcomma
 (script-stencil feta "laltcomma" . "raltcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
 (circulus
 (script-stencil feta "circulus" . "circulus")
 (side-relative-direction . -1)
 (avoid-slur . ignore)
 (padding . 0.2)
 (quantize-position . #t)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
 (coda (script-stencil feta "coda" . "coda")
 (padding . 0.2)
 (avoid-slur . outside))
```

```

 (side-axis . 1)
 (direction . 1))
(commma (script-stencil feta "lcomma" . "rcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(downbow
 (script-stencil feta "ddownbow" . "udownbow")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(downmordent
 (script-stencil
 feta
 "downmordent"
 .
 "downmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(downprall
 (script-stencil feta "downprall" . "downprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(espressivo
 (avoid-slur . around)
 (padding . 0.2)
 (script-stencil feta "espr" . "espr")
 (side-axis . 1)
 (side-relative-direction . -1))
(fermata
 (script-stencil feta "dfermata" . "ufermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(flageolet
 (script-stencil feta "flageolet" . "flageolet")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)

```

```

 (script-priority . 50))
(halfopen
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "halfopen" . "halfopen")
 (side-axis . 1)
 (direction . 1))
(halfopenvertical
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil
 feta
 "halfopenvertical"
 .
 "halfopenvertical")
 (side-axis . 1)
 (direction . 1))
(haydnturn
 (script-stencil feta "haydnturn" . "haydnturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))
(henzelongfermata
 (script-stencil
 feta
 "dhenzelongfermata"
 .
 "uhenzelongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(henzeshortfermata
 (script-stencil
 feta
 "dhenzeshortfermata"
 .
 "uhenzeshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ictus (script-stencil feta "ictus" . "ictus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)

```

```

 (script-priority . -100)
 (side-axis . 1)
 (direction . -1))
(lheel (script-stencil feta "upedalheel" . "upedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(lineprall
 (script-stencil feta "lineprall" . "lineprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(longfermata
 (script-stencil
 feta
 "dlongfermata"
 .
 "ulongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ltoe (script-stencil feta "upedaltoe" . "upedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(marcato
 (script-stencil feta "dmarcato" . "umarcato")
 (padding . 0.2)
 (avoid-slur . inside)
 (quantize-position . #t)
 (side-axis . 1)
 (side-relative-direction . -1))
(mordent
 (script-stencil feta "mordent" . "mordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(open (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "open" . "open")
 (side-axis . 1)
 (direction . 1))
(outsidecomma
 (avoid-slur . around)
 (direction . 1))

```

```

(padding . 0.2)
(side-axis . 1)
(script-stencil feta "lcomma" . "rcomma"))
(portato
 (script-stencil feta "uportato" . "dportato")
 (avoid-slur . around)
 (padding . 0.45)
 (side-axis . 1)
 (side-relative-direction . -1))
(prall (script-stencil feta "prall" . "prall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(pralldown
 (script-stencil feta "pralldown" . "pralldown")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallmordent
 (script-stencil
 feta
 "prallmordent"
 .
 "prallmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallprall
 (script-stencil feta "prallprall" . "prallprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallup
 (script-stencil feta "prallup" . "prallup")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(reverseturn
 (script-stencil
 feta
 "reverseturn"
 .
 "reverseturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))

```

```

(rheel (script-stencil feta "dpedalheel" . "dpedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(rtoe (script-stencil feta "dpedaltoe" . "dpedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(segno (script-stencil feta "segno" . "segno")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(semicirculus
 (script-stencil
 feta
 "dsemicirculus"
 .
 "dsemicirculus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
(shortfermata
 (script-stencil
 feta
 "dshortfermata"
 .
 "ushortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(signumcongruentiae
 (script-stencil
 feta
 "dsignumcongruentiae"
 .
 "usignumcongruentiae")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(slashturn
 (script-stencil feta "slashturn" . "slashturn"))

```



```

(padding . 0.2)
(avoid-slur . inside)
(side-axis . 1)
(direction . 1))
(snappizzicato
 (script-stencil
 feta
 "snappizzicato"
 .
 "snappizzicato")
(padding . 0.2)
(avoid-slur . outside)
(side-axis . 1)
(direction . 1))
(staccatissimo
 (avoid-slur . inside)
 (quantize-position . #t)
 (script-stencil
 feta
 "dstaccatissimo"
 .
 "ustaccatissimo")
(padding . 0.2)
(skyline-horizontal-padding . 0.1)
(side-axis . 1)
(side-relative-direction . -1)
(toward-stem-shift . 1.0)
(toward-stem-shift-in-column . 0.0))
(staccato
 (script-stencil feta "staccato" . "staccato")
 (side-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
(tenuto
 (script-stencil feta "tenuto" . "tenuto")
 (quantize-position . #t)
 (avoid-slur . inside)
 (padding . 0.2)
 (script-priority . -50)
 (side-axis . 1)

```

```

 (side-relative-direction . -1))
(trill (script-stencil feta "trill" . "trill")
 (direction . 1)
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (script-priority . 150))
(turn (script-stencil feta "turn" . "turn")
 (avoid-slur . inside)
 (padding . 0.2)
 (side-axis . 1)
 (direction . 1))
(upbow (script-stencil feta "dupbow" . "uupbow")
 (avoid-slur . around)
 (padding . 0.2)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(upmordent
 (script-stencil feta "upmordent" . "upmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(upprall
 (script-stencil feta "upprall" . "upprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(varcoda
 (script-stencil feta "varcoda" . "varcoda")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(varcomma
 (script-stencil feta "lvarcomma" . "rvarcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(verylongfermata
 (script-stencil
 feta
 "dverylongfermata"
 .
 "uverylongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75))

```

```

 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
 (veryshortfermata
 (script-stencil
 feta
 "dveryshortfermata"
 .
 "uveryshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (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 "/".
- Set context property soloIIIText 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 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 4))))
((9 . 4)
 (beamExceptions (end (1/32 8 8 8 8 8 8 8 8))))
((12 . 4)
 (beamExceptions
 (end (1/32 8 8 8 8 8 8 8 8 8 8 8 8))))
((5 . 8) (beatStructure 3 2))
((8 . 8) (beatStructure 3 3 2)))

```

- Set context property timing to #f.
- Set context property timing to #t.
- Set context property topLevelAlignment to #t.
- Set context property underlyingRepeatBarType to "||".
- Set grob property common-shortest-duration in SpacingSpanner (page 693), 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 349).

Context `StandaloneRhythmScore` can contain `ChoirStaff` (page 67), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `StandaloneRhythmStaff` (page 349), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Bar_number_engraver` (page 456)

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 `Staff_collecting_engraver` (page 499). 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 first volta number for the current \alternative element.  
Not set outside of alternatives.

`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)

The current point within the measure.

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): `BarNumber` (page 542), and `CenteredBarNumber` (page 561).

`Beam_collision_engraver` (page 457)

Help beams avoid colliding with notes and clefs in other voices.

`Break_align_engraver` (page 460)

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 554), `BreakAlignment` (page 555), and `LeftEdge` (page 632).

`Centered_bar_number_align_engraver` (page 461)

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 562).

`Concurrent_hairpin_engraver` (page 465)

Collect concurrent hairpins.

`Footnote_engraver` (page 473)

Create footnote texts.

This engraver creates the following layout object(s): `Footnote` (page 606).

`Grace_spacing_engraver` (page 475)

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 611).

`Jump_engraver` (page 478)

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 499), also needs to be there so that marks appear at the intended Y location.

Music types accepted: `ad-hoc-jump-event` (page 49), `dal-segno-event` (page 51), and `fine-event` (page 52),

## 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 621).

`Mark_engraver` (page 482)

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 499), 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 483). 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 570), `RehearsalMark` (page 673), `SectionLabel` (page 681), and `SegnoMark` (page 683).

`Mark_tracking_translator` (page 483)

This translator chooses which marks `Mark_engraver` should engrave.

Music types accepted: `ad-hoc-mark-event` (page 49), `coda-mark-event` (page 51), `rehearsal-mark-event` (page 56), `section-label-event` (page 57), and `segno-mark-event` (page 57),

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 485)

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 `Staff_collecting_engraver` (page 499).

Music types accepted: `tempo-change-event` (page 59),

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 647).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Paper_column_engraver` (page 490)

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 50), and `label-event` (page 53),

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 655), and `PaperColumn` (page 665).

`Parenthesis_engraver` (page 491)

Parenthesize objects whose `parenthesize` property is `#t`.

This engraver creates the following layout object(s): `Parentheses` (page 666).

`Repeat_acknowledge_engraver` (page 494)

This translator adds entries to `repeatCommands` for events generated by `\\repeat volta`.

Music types accepted: `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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 496)

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 575), and `ControlPolygon` (page 576).

`Spacing_engraver` (page 498)

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted: `spacing-section-event` (page 57),

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` (non-negative exact rational or `+inf.0`)

Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s): `SpacingSpanner` (page 693).

`Spanner_tracking_engraver` (page 499)

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 499)

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 500)

This engraver ensures that stanza numbers are neatly aligned.

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Text_mark_engraver` (page 503)

Engraves arbitrary textual marks.

Music types accepted: `text-mark-event` (page 59),

Properties (read)

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): `TextMark` (page 719).

`Timing_translator` (page 505)

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 49), `bar-check-event` (page 50), `bar-event` (page 50), `fine-event` (page 52), and `partial-event` (page 55),

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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`timeSignatureFraction` (simple time signature)

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 first volta number for the current `\alternative` element. Not set outside of alternatives.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`timeSignatureFraction` (simple time signature)

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

`Tweak_engraver` (page 507)

Read the tweaks property from the originating event, and set properties.

`Vertical_align_engraver` (page 507)

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 698), and `VerticalAlignment` (page 742).

`Volta_engraver` (page 507)

Make volta brackets.

Music types accepted: `dal-segno-event` (page 51), `fine-event` (page 52), and `volta-span-event` (page 60),

Properties (read)

`currentCommandColumn` (graphical (layout) object)

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

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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.

This engraver creates the following layout object(s): `VoltaBracket` (page 745), and `VoltaBracketSpanner` (page 747).

### 2.1.35 StandaloneRhythmStaff

A Staff-level context for use by `\markup \rhythm`.

This context also accepts commands for the following context(s): `Staff` (page 309), and `Staff` (page 309).

This context creates the following layout object(s): `BarLine` (page 538), `BreathingSign` (page 557), `CaesuraScript` (page 559), `DotColumn` (page 588), `InstrumentName` (page 618), `LedgerLineSpanner` (page 631), `StaffHighlight` (page 699), `StaffSpacing` (page 700), `StaffSymbol` (page 701), and `VerticalAxisGroup` (page 742).

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 701), to 0.
- Set grob property `line-count` in `StaffSymbol` (page 701), to 1.
- Set grob property `neutral-direction` in `Beam` (page 549), to 1.
- Set grob property `neutral-direction` in `Stem` (page 703), to 1.
- Set grob property `staff-padding` in `VoltaBracket` (page 745), to 3.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `StandaloneRhythmVoice` (page 354).

Context `StandaloneRhythmStaff` can contain `CueVoice` (page 100), `NullVoice` (page 247), `StandaloneRhythmVoice` (page 354), and `Voice` (page 439).

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

`Axis_group_engraver` (page 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)  
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)  
 Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*  
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*  
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

`Dot_column_engraver` (page 467)

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 588).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Instrument_name_engraver` (page 477)

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 618).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

Output\_property\_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Pitch\_squash\_engraver (page 493)

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 496)

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 700).

Staff\_highlight\_engraver (page 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff\_symbol\_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

### 2.1.36 StandaloneRhythmVoice

A Voice-level context for use by \markup \rhythm.

This context also accepts commands for the following context(s): Voice (page 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), FingerGlideSpanner (page 601), Fingering (page 603), Flag (page 605), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), LigatureBracket (page 634), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653),

NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), and VoiceFollower (page 744).

This context sets the following properties:

- Set grob property direction in Stem (page 703), 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 452)

Generate an Arpeggio symbol.

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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto\_beam\_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem\_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

**Beam\_engraver** (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

**Bend\_engraver** (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

**Breathing\_sign\_engraver** (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

**Chord\_tremolo\_engraver** (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster\_spanner\_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots\_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double\_percent\_repeat\_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Dynamic\_align\_engraver (page 469)

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 595).

Dynamic\_engraver (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 49),

break-dynamic-span-event (page 50), and span-dynamic-event (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two `Fingering`, `StringNumber` or `StrokeFinger` grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Fingering_engraver` (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

`Font_size_engraver` (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

Properties (read)

`glissandoMap` (list)

A map in the form of ‘((source1 . target1) (source2 . target2) (source . 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 610).

Grace\_auto\_beam\_engraver (page 474)

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 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_beam\_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_engraver (page 475)

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 476)

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 478)
- 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 619).
- `Laissez_vibrer_engraver` (page 481)
- Create laissez vibrer items.
  - Music types accepted: `laissez-vibrer-event` (page 53),
  - This engraver creates the following layout object(s): `LaissezVibrerTie` (page 629), and `LaissezVibrerTieColumn` (page 630).
- `Ligature_bracket_engraver` (page 481)
- Handle `Ligature_events` by engraving `Ligature` brackets.
  - Music types accepted: `ligature-event` (page 53),
  - This engraver creates the following layout object(s): `LigatureBracket` (page 634).
- `Multi_measure_rest_engraver` (page 487)
- Engrave multi-measure rests that are produced with ‘R’. It reads `measureStartNow` and `internalBarNumber` to determine what number to print over the `MultiMeasureRest` (page 649).
  - Music types accepted: `multi-measure-articulation-event` (page 54), `multi-measure-rest-event` (page 54), and `multi-measure-text-event` (page 54),
  - 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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).
- `New_fingering_engraver` (page 487)
- 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 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

`Note_head_line_engraver` (page 488)

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 744).

`Note_heads_engraver` (page 488)

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

## 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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

`Repeat_tie_engraver` (page 494)

Create repeat ties.

Music types accepted: `repeat-tie-event` (page 56),

This engraver creates the following layout object(s): `RepeatTie` (page 676), and `RepeatTieColumn` (page 677).

`Rest_engraver` (page 495)

Engrave rests.

Music types accepted: `rest-event` (page 56),

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 678).

`Rhythmic_column_engraver` (page 495)

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): `NoteColumn` (page 657).

`Script_column_engraver` (page 495)

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 680).

`Script_engraver` (page 496)

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 50),

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 679).

`Slash_repeat_engraver` (page 497)

Make beat repeats.

Music types accepted: `repeat-slash-event` (page 56),

This engraver creates the following layout object(s): `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

`Slur_engraver` (page 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

`Spanner_break_forbid_engraver` (page 499)

Forbid breaks in certain spanners.

`Stem_engraver` (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and `TieColumn` (page 727).

`Trill_spanner_engraver` (page 506)

Create trill spanners.

Music types accepted: `trill-span-event` (page 60),

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 734).

`Tuplet_engraver` (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

### 2.1.37 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 309).

This context creates the following layout object(s): `BarLine` (page 538), `BassFigure` (page 544), `BassFigureAlignment` (page 544), `BassFigureAlignmentPositioning` (page 545), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), `BassFigureLine` (page 548), `BreathingSign` (page 557), `CaesuraScript` (page 559), `Clef` (page 565), `ClefModifier` (page 568), `CueClef` (page 577), `CueEndClef` (page 580), `DotColumn` (page 588), `FingeringColumn` (page 605), `InstrumentName` (page 618), `LedgerLineSpanner` (page 631), `NoteCollision` (page 656), `OptionalMaterialBracket` (page 661), `PianoPedalBracket` (page 672), `RestCollision` (page 679), `ScriptColumn` (page 680), `ScriptRow` (page 681), `SostenutoPedal` (page 690), `SostenutoPedalLineSpanner` (page 692), `StaffEllipsis` (page 696), `StaffHighlight` (page 699), `StaffSpacing` (page 700), `StaffSymbol` (page 701), `SustainPedal` (page 710), `SustainPedalLineSpanner` (page 711), `TimeSignature` (page 727), `UnaCordaPedal` (page 739), `UnaCordaPedalLineSpanner` (page 740), and `VerticalAxisGroup` (page 742).

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 `avoid-note-head` in `Stem` (page 703), to `#t`.
- Set grob property `beam-thickness` in `Beam` (page 549), to `0.32`.
- Set grob property `beam-thickness` in `StemTremolo` (page 705), to `0.32`.
- Set grob property `beam-width` in `StemTremolo` (page 705), to `stem-tremolo::calc-tab-width`.
- Set grob property `bound-details.left` in `Glissando` (page 610), to :  
`'((attach-dir . 1) (padding . 0.3))`
- Set grob property `bound-details.right` in `Glissando` (page 610), to :  
`'((attach-dir . -1) (padding . 0.3))`
- Set grob property `control-points` in `Slur` (page 688), to `#<unpure-pure-container  
#<procedure at lily/music-functions.scm:2585:16 (grob . rest)>>`.
- Set grob property `details` in `Stem` (page 703), to :  
`'((lengths 0 0 0 0 0 0)  
  (beamed-lengths 0 0 0)  
  (beamed-minimum-free-lengths 0 0 0)  
  (beamed-extreme-minimum-free-lengths 0 0)  
  (stem-shorten 0 0))`
- Set grob property `extra-dy` in `Glissando` (page 610), to `glissando::calc-tab-extra-dy`.
- Set grob property `glyph-name` in `TabNoteHead` (page 717), to `tab-note-head::calc-glyph-name`.
- Set grob property `ignore-collision` in `NoteColumn` (page 657), to `#t`.
- Set grob property `length-fraction` in `Beam` (page 549), to `0.62`.
- Set grob property `length-fraction` in `StemTremolo` (page 705), to `#<procedure at  
ice-9/eval.scm:333:13 (a)>`.
- Set grob property `no-stem-extend` in `Stem` (page 703), to `#t`.
- Set grob property `staff-space` in `StaffSymbol` (page 701), to `1.5`.
- Set grob property `stencil` in `Arpeggio` (page 535), to `#f`.
- Set grob property `stencil` in `Beam` (page 549), to `#f`.
- Set grob property `stencil` in `Clef` (page 565), to `clef::print-modern-tab-if-set`.
- Set grob property `stencil` in `Dots` (page 588), to `#f`.
- Set grob property `stencil` in `DynamicTextSpanner` (page 599), to `#f`.
- Set grob property `stencil` in `DynamicText` (page 597), to `#f`.
- Set grob property `stencil` in `Flag` (page 605), to `#f`.
- Set grob property `stencil` in `Glissando` (page 610), to `glissando::draw-tab-glissando`.
- Set grob property `stencil` in `Hairpin` (page 614), to `#f`.
- Set grob property `stencil` in `LaissezVibrerTie` (page 629), to `#f`.
- Set grob property `stencil` in `MultiMeasureRestNumber` (page 650), to `#f`.
- Set grob property `stencil` in `MultiMeasureRestScript` (page 652), to `#f`.
- Set grob property `stencil` in `MultiMeasureRestText` (page 653), to `#f`.
- Set grob property `stencil` in `MultiMeasureRest` (page 649), to `#f`.

- Set grob property stencil in PhrasingSlur (page 670), to #f.
- Set grob property stencil in RepeatTie (page 676), to #f.
- Set grob property stencil in Rest (page 678), to #f.
- Set grob property stencil in Script (page 679), to #f.
- Set grob property stencil in StemTremolo (page 705), to #f.
- Set grob property stencil in Stem (page 703), to #f.
- Set grob property stencil in TabNoteHead (page 717), to `tab-note-head::whiteout-if-style-set`.
- Set grob property stencil in TextScript (page 721), to #f.
- Set grob property stencil in TextSpanner (page 723), to #f.
- Set grob property stencil in Tie (page 725), to #f.
- Set grob property stencil in TimeSignature (page 727), to #f.
- Set grob property stencil in TupletBracket (page 736), to #f.
- Set grob property stencil in TupletNumber (page 737), to #f.
- Set grob property style in Flag (page 605), 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 377).

Context TabStaff can contain CueVoice (page 100), NullVoice (page 247), and TabVoice (page 377).

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

Alteration\_glyph\_engraver (page 451)

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 453)

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 742).



Bar\_engraver (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)  
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)  
 Notate volta-style repeats even when the repeat count is 1.

`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 return-count`  
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`  
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Caesura_engraver` (page 460)

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 51),

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 557), and `CaesuraScript` (page 559).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.
- `cueClefTranspositionStyle` (symbol)  
Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

`Horizontal_script_engraver` (page 477)

Aligns Script horizontally

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.

`Instrument_name_engraver` (page 477)

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 618).

Ledger\_line\_engraver (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

Merge\_mmrest\_numbers\_engraver (page 485)

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 488)

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 680).

Optional\_material\_bracket\_engraver (page 489)

Notate in-staff brackets for optional material.

Music types accepted: optional-material-event (page 55),

This engraver creates the following layout object(s): OptionalMaterialBracket (page 661).

Output\_property\_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Piano\_pedal\_align\_engraver (page 492)

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 692), SustainPedalLineSpanner (page 711), and UnaCordaPedalLineSpanner (page 740).

Piano\_pedal\_engraver (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 57), sustain-event (page 59), and una-corda-event (page 60),

## 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 672), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `UnaCordaPedal` (page 739).

`Pure_from_neighbor_engraver` (page 494)

Coordinates items that get their pure heights from their neighbors.

`Rest_collision_engraver` (page 495)

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 679).

`Script_row_engraver` (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): `ScriptRow` (page 681).

`Separating_line_group_engraver` (page 496)

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 700).

`Skip_typesetting_engraver` (page 497)

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 696).

`Staff_collecting_engraver` (page 499)

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 499)

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_symbol_engraver` (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Tab_staff_symbol_engraver` (page 502)

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 701).

`Time_signature_engraver` (page 504)

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

timeSignatureFraction (simple time signature)

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 727).

### 2.1.38 TabVoice

The voice context used within a TabStaff context. Usually left to be created implicitly.

This context also accepts commands for the following context(s): Voice (page 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BendSpanner (page 552), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), FingerGlideSpanner (page 601), Flag (page 605), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), LigatureBracket (page 634), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), Slur (page 688), Stem (page 703), StemStub (page 705), StemTremolo (page 705), TabNoteHead (page 717), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), and VoiceFollower (page 744).

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 452)

Generate an Arpeggio symbol.

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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto\_beam\_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem\_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Beam_engraver` (page 457)

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

Music types accepted: `beam-event` (page 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Bend_engraver` (page 459)

Create fall spanners.

Music types accepted: `bend-after-event` (page 50),

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 551).

Bend\_spanner\_engraver (page 459)

Engraver to print a BendSpanner.

Music types accepted: bend-span-event (page 50), note-event (page 55), and string-number-event (page 58),

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 552).

Breathing\_sign\_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

Chord\_tremolo\_engraver (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster\_spanner\_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots\_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double\_percent\_repeat\_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`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 590), and `DoublePercentRepeatCounter` (page 591).

`Dynamic_align_engraver` (page 469)

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 595).

`Dynamic_engraver` (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49), `break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Finger_glide_engraver` (page 472)

Engraver to print a line between two Fingering, StringNumber or StrokeFinger grobs.

Music types accepted: `note-event` (page 55),

This engraver creates the following layout object(s): `FingerGlideSpanner` (page 601).

`Font_size_engraver` (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

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 610).

`Grace_auto_beam_engraver` (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

`Grace_beam_engraver` (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_engraver (page 475)

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 476)

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 478)

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 619).

Laissez\_vibrer\_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

Ligature\_bracket\_engraver (page 481)

Handle Ligature\_events by engraving Ligature brackets.

Music types accepted: ligature-event (page 53),

This engraver creates the following layout object(s): LigatureBracket (page 634).

Multi\_measure\_rest\_engraver (page 487)

Engrave multi-measure rests that are produced with ‘R’. It reads measureStartNow and internalBarNumber to determine what number to print over the MultiMeasureRest (page 649).

Music types accepted: multi-measure-articulation-event (page 54), multi-measure-rest-event (page 54), and multi-measure-text-event (page 54),

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 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), and MultiMeasureRestText (page 653).

Note\_head\_line\_engraver (page 488)

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 744).

Note\_spacing\_engraver (page 489)

Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 660).

Output\_property\_engraver (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

Part\_combine\_engraver (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: note-event (page 55), and part-combine-event (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Repeat_tie_engraver` (page 494)

Create repeat ties.

Music types accepted: `repeat-tie-event` (page 56),

This engraver creates the following layout object(s): `RepeatTie` (page 676), and `RepeatTieColumn` (page 677).

`Rest_engraver` (page 495)

Engrave rests.

Music types accepted: `rest-event` (page 56),

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 678).

**Rhythmic\_column\_engraver** (page 495)

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

**Script\_column\_engraver** (page 495)

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 680).

**Script\_engraver** (page 496)

Handle note scripted articulations.

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

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 679).

**Slash\_repeat\_engraver** (page 497)

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

**Slur\_engraver** (page 497)

Build slur grobs from slur events.

Music types accepted: note-event (page 55), and slur-event (page 57),

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 688).

**Spanner\_break\_forbid\_engraver** (page 499)

Forbid breaks in certain spanners.

**Stem\_engraver** (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: tremolo-event (page 59),

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 605), Stem (page 703), StemStub (page 705), and StemTremolo (page 705).

Tab\_note\_heads\_engraver (page 501)

Generate one or more tablature note heads from event of type NoteEvent.

Music types accepted: fingering-event (page 52), note-event (page 55), and string-number-event (page 58),

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 717).

Tab\_tie\_follow\_engraver (page 502)

Adjust TabNoteHead properties when the TabNoteHead holds a RepeatTie, when a Tie ends and when a Slur or Glissando starts at a tied TabNoteHead.

Properties (read)

- `tabFullNotation` (boolean)  
Flag whether `\tabFullNotation` is used

`Text_engraver` (page 503)  
Create text scripts.  
Music types accepted: `text-script-event` (page 59),  
This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)  
Create text spanner from an event.  
Music types accepted: `text-span-event` (page 59),  
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 723).

`Tie_engraver` (page 503)  
Generate ties between note heads of equal pitch.  
Music types accepted: `tie-event` (page 59),  
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 725), and `TieColumn` (page 727).

`Trill_spanner_engraver` (page 506)  
Create trill spanners.  
Music types accepted: `trill-span-event` (page 60),  
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 734).

`Tuplet_engraver` (page 506)  
Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

### 2.1.39 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 218).

This context creates the following layout object(s): `InstrumentName` (page 618), `LyricExtender` (page 635), `LyricHyphen` (page 636), `LyricSpace` (page 639), `LyricText` (page 640), `StanzaNumber` (page 702), `VerticalAxisGroup` (page 742), and `VowelTransition` (page 748).

This context sets the following properties:

- Set context property `instrumentName` to `'()`.
- Set context property `lyricRepeatCountFormatter` to `#<procedure at lily/translation-functions.scm:214:4 (context repeat-count)>`.
- Set context property `searchForVoice` to `#f`.
- Set context property `shortInstrumentName` to `'()`.
- Set grob property `bar-extent` in `BarLine` (page 538), to :  
`'(-0.05 . 0.05)`
- Set grob property `font-series` in `LyricHyphen` (page 636), to `'normal`.
- Set grob property `font-size` in `InstrumentName` (page 618), to `1.0`.
- Set grob property `font-size` in `LyricHyphen` (page 636), to `-4`.
- Set grob property `font-size` in `LyricText` (page 640), to `-4`.
- Set grob property `nonstaff-nonstaff-spacing` in `VerticalAxisGroup` (page 742), to :  
`'((basic-distance . 0)  
  (minimum-distance . 2.8)  
  (padding . 0.2)  
  (stretchability . 0))`
- Set grob property `nonstaff-relatedstaff-spacing` in `VerticalAxisGroup` (page 742), to :  
`'((basic-distance . 5.5)  
  (padding . 0.5)  
  (stretchability . 1))`
- Set grob property `nonstaff-unrelatedstaff-spacing.padding` in `VerticalAxisGroup` (page 742), to `1.5`.
- Set grob property `remove-empty` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `remove-first` in `VerticalAxisGroup` (page 742), to `#t`.
- Set grob property `self-alignment-Y` in `InstrumentName` (page 618), to `#f`.

- Set grob property `short-bar-extent` in `BarLine` (page 538), to :  
'(-0.05 . 0.05)
- Set grob property `staff-affinity` in `VerticalAxisGroup` (page 742), to 1.
- Set grob property `stencil` in `LyricHyphen` (page 636), 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 453)

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 742).

`Extender_engraver` (page 471)

Create lyric extenders.

Music types accepted: `completize-extender-event` (page 51), and `extender-event` (page 52),

Properties (read)

`extendersOverRests` (boolean)

Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): `LyricExtender` (page 635).

`Font_size_engraver` (page 472)

Put `fontSize` into `font-size` grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Hyphen_engraver` (page 477)

Create lyric hyphens, vowel transitions and distance constraints between words.

Music types accepted: `hyphen-event` (page 53), and `vowel-transition-event` (page 60),

This engraver creates the following layout object(s): `LyricHyphen` (page 636), `LyricSpace` (page 639), and `VowelTransition` (page 748).

Instrument\_name\_engraver (page 477)

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 618).

Lyric\_engraver (page 481)

Engrave text for lyrics.

Music types accepted: lyric-event (page 53),

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 640).

Pure\_from\_neighbor\_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Stanza\_number\_engraver (page 500)

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 702).

### 2.1.40 VaticanaScore

Top-level context replacing Score for Gregorian chant notated in Vaticana style. Compared to Score, it changes the staff line color to red, uses packed spacing, and removes bar numbers.

This context also accepts commands for the following context(s): Score (page 283), and Timing (page 283).

This context creates the following layout object(s): BreakAlignGroup (page 554), BreakAlignment (page 555), CenteredBarNumberLineSpanner (page 562), CodaMark (page 570), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), GraceSpacing (page 611), JumpScript (page 621), LeftEdge (page 632), MetronomeMark (page 647), NonMusicalPaperColumn (page 655), PaperColumn (page 665), Parentheses (page 666), RehearsalMark (page 673), SectionLabel (page 681), SegnoMark (page 683), SpacingSpanner (page 693), StaffGrouper (page 698), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TextMark (page 719), VerticalAlignment (page 742), VoltaBracket (page 745), and VoltaBracketSpanner (page 747).

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
 measureStartNow
 lastChord)
```
- Set context property associatedVoiceType to 'Voice.
- Set context property autoAccidentals to:
 

```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```
- 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-
- Set context property beamHalfMeasure to #t.
- Set context property breathMarkDefinitions to:
 

```
'((altcomma
 (text #<procedure musicglyph-markup (layout props glyph-name)>
 "scripts.raltcomma"))
 (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
 (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.uupbow"))
(varcomma
 (text #<procedure musicglyph-markup (layout props glyph-name)>
 "scripts.rvarcomma")))

```

- 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 line-markup (layout props args)>
 ((#<procedure fontsize-markup (layout props increment arg)>

```

```

 "•")))))
((#<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 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 at lily/translation-functions.scm:228:4 (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>.
- 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)
 (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 "m".
- Set context property noChordSymbol to "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 `printTrivialVoltaRepeats` to `#f`.
- 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 at lily/translation-functions.scm:228:4 (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-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
 (altcomma
 (script-stencil feta "laltcomma" . "raltcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
 (circulus
 (script-stencil feta "circulus" . "circulus")
 (side-relative-direction . -1)
 (avoid-slur . ignore)
 (padding . 0.2)
 (quantize-position . #t)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
 (coda (script-stencil feta "coda" . "coda")
 (padding . 0.2)
 (avoid-slur . outside))
```

```

 (side-axis . 1)
 (direction . 1))
(commma (script-stencil feta "lcomma" . "rcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(downbow
 (script-stencil feta "ddownbow" . "udownbow")
 (padding . 0.2)
 (skyline-horizontal-padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(downmordent
 (script-stencil
 feta
 "downmordent"
 .
 "downmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(downprall
 (script-stencil feta "downprall" . "downprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(espressivo
 (avoid-slur . around)
 (padding . 0.2)
 (script-stencil feta "espr" . "espr")
 (side-axis . 1)
 (side-relative-direction . -1))
(fermata
 (script-stencil feta "dfermata" . "ufermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(flageolet
 (script-stencil feta "flageolet" . "flageolet")
 (padding . 0.2)
 (avoid-slur . around)
 (direction . 1)
 (side-axis . 1)

```

```

 (script-priority . 50))
(halfopen
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "halfopen" . "halfopen")
 (side-axis . 1)
 (direction . 1))
(halfopenvertical
 (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil
 feta
 "halfopenvertical"
 .
 "halfopenvertical")
 (side-axis . 1)
 (direction . 1))
(haydnturn
 (script-stencil feta "haydnturn" . "haydnturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))
(henzelongfermata
 (script-stencil
 feta
 "dhenzelongfermata"
 .
 "uhenzelongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(henzeshortfermata
 (script-stencil
 feta
 "dhenzeshortfermata"
 .
 "uhenzeshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ictus (script-stencil feta "ictus" . "ictus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2))

```



```

 (script-priority . -100)
 (side-axis . 1)
 (direction . -1))
(lheel (script-stencil feta "upedalheel" . "upedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(lineprall
 (script-stencil feta "lineprall" . "lineprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(longfermata
 (script-stencil
 feta
 "dlongfermata"
 .
 "ulongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(ltoe (script-stencil feta "upedaltoe" . "upedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . -1))
(marcato
 (script-stencil feta "dmarcato" . "umarcato")
 (padding . 0.2)
 (avoid-slur . inside)
 (quantize-position . #t)
 (side-axis . 1)
 (side-relative-direction . -1))
(mordent
 (script-stencil feta "mordent" . "mordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(open (avoid-slur . outside)
 (padding . 0.2)
 (script-stencil feta "open" . "open")
 (side-axis . 1)
 (direction . 1))
(outsidecomma
 (avoid-slur . around)
 (direction . 1))

```

```

(padding . 0.2)
(side-axis . 1)
(script-stencil feta "lcomma" . "rcomma"))
(portato
 (script-stencil feta "uportato" . "dportato")
 (avoid-slur . around)
 (padding . 0.45)
 (side-axis . 1)
 (side-relative-direction . -1))
(prall (script-stencil feta "prall" . "prall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(pralldown
 (script-stencil feta "pralldown" . "pralldown")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallmordent
 (script-stencil
 feta
 "prallmordent"
 .
 "prallmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallprall
 (script-stencil feta "prallprall" . "prallprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(prallup
 (script-stencil feta "prallup" . "prallup")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(reverseturn
 (script-stencil
 feta
 "reverseturn"
 .
 "reverseturn")
 (padding . 0.2)
 (avoid-slur . inside)
 (side-axis . 1)
 (direction . 1))

```

```

(rheel (script-stencil feta "dpedalheel" . "dpedalheel")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(rtoe (script-stencil feta "dpedaltoe" . "dpedaltoe")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(segno (script-stencil feta "segno" . "segno")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(semicirculus
 (script-stencil
 feta
 "dsemicirculus"
 .
 "dsemicirculus")
 (side-relative-direction . -1)
 (quantize-position . #t)
 (avoid-slur . ignore)
 (padding . 0.2)
 (script-priority . -100)
 (side-axis . 1)
 (direction . 1))
(shortfermata
 (script-stencil
 feta
 "dshortfermata"
 .
 "ushortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
(signumcongruentiae
 (script-stencil
 feta
 "dsignumcongruentiae"
 .
 "usignumcongruentiae")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(slashturn
 (script-stencil feta "slashturn" . "slashturn"))

```

```

(padding . 0.2)
(avoid-slur . inside)
(side-axis . 1)
(direction . 1))
(snappizzicato
 (script-stencil
 feta
 "snappizzicato"
 .
 "snappizzicato")
(padding . 0.2)
(avoid-slur . outside)
(side-axis . 1)
(direction . 1))
(staccatissimo
 (avoid-slur . inside)
 (quantize-position . #t)
 (script-stencil
 feta
 "dstaccatissimo"
 .
 "ustaccatissimo")
(padding . 0.2)
(skyline-horizontal-padding . 0.1)
(side-axis . 1)
(side-relative-direction . -1)
(toward-stem-shift . 1.0)
(toward-stem-shift-in-column . 0.0))
(staccato
 (script-stencil feta "staccato" . "staccato")
 (side-axis . 1)
 (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)
 (side-axis . 1)
 (direction . 1))
(tenuto
 (script-stencil feta "tenuto" . "tenuto")
 (quantize-position . #t)
 (avoid-slur . inside)
 (padding . 0.2)
 (script-priority . -50)
 (side-axis . 1)

```

```

 (side-relative-direction . -1))
(trill (script-stencil feta "trill" . "trill")
 (direction . 1)
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (script-priority . 150))
(turn (script-stencil feta "turn" . "turn")
 (avoid-slur . inside)
 (padding . 0.2)
 (side-axis . 1)
 (direction . 1))
(upbow (script-stencil feta "dupbow" . "uupbow")
 (avoid-slur . around)
 (padding . 0.2)
 (direction . 1)
 (side-axis . 1)
 (script-priority . 180))
(upmordent
 (script-stencil feta "upmordent" . "upmordent")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(upprall
 (script-stencil feta "upprall" . "upprall")
 (padding . 0.2)
 (avoid-slur . around)
 (side-axis . 1)
 (direction . 1))
(varcoda
 (script-stencil feta "varcoda" . "varcoda")
 (padding . 0.2)
 (avoid-slur . outside)
 (side-axis . 1)
 (direction . 1))
(varcomma
 (script-stencil feta "lvarcomma" . "rvarcomma")
 (quantize-position . #t)
 (padding . 0.2)
 (avoid-slur . ignore)
 (side-axis . 1)
 (direction . 1))
(verylongfermata
 (script-stencil
 feta
 "dverylongfermata"
 .
 "uverylongfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75))

```

```

 (script-priority . 175)
 (side-axis . 1)
 (direction . 1))
 (veryshortfermata
 (script-stencil
 feta
 "dveryshortfermata"
 .
 "uveryshortfermata")
 (padding . 0.4)
 (avoid-slur . around)
 (outside-staff-priority . 75)
 (script-priority . 175)
 (side-axis . 1)
 (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 "/".
- Set context property soloIIIText 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 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 4))))
((9 . 4)
 (beamExceptions (end (1/32 8 8 8 8 8 8 8 8))))
((12 . 4)
 (beamExceptions
 (end (1/32 8 8 8 8 8 8 8 8 8 8 8 8))))
((5 . 8) (beatStructure 3 2))
((8 . 8) (beatStructure 3 3 2)))

```

- Set context property timing to #f.
- Set context property timing to #t.
- Set context property topLevelAlignment to #t.
- Set context property underlyingRepeatBarType to "||".
- Set grob property color in LedgerLineSpanner (page 631), to :  
'(1.0 0.0 0.0)
- Set grob property color in StaffSymbol (page 701), to :  
'(1.0 0.0 0.0)
- Set grob property packed-spacing in SpacingSpanner (page 693), to #t.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `VaticanaStaff` (page 414).

Context `VaticanaScore` can contain `ChoirStaff` (page 67), `ChordNames` (page 98), `Devnull` (page 110), `DrumStaff` (page 111), `Dynamics` (page 130), `FiguredBass` (page 135), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `OneStaff` (page 249), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

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

`Beam_collision_engraver` (page 457)

Help beams avoid colliding with notes and clefs in other voices.

`Break_align_engraver` (page 460)

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 554), `BreakAlignment` (page 555), and `LeftEdge` (page 632).

Centered\_bar\_number\_align\_engraver (page 461)

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 562).

Concurrent\_hairpin\_engraver (page 465)

Collect concurrent hairpins.

Footnote\_engraver (page 473)

Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 606).

Grace\_spacing\_engraver (page 475)

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 611).

Jump\_engraver (page 478)

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 499), also needs to be there so that marks appear at the intended Y location.

Music types accepted: ad-hoc-jump-event (page 49), dal-segno-event (page 51), and fine-event (page 52),

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 621).

`Mark_engraver` (page 482)

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 499), 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 483). 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 570), RehearsalMark (page 673), SectionLabel (page 681), and SegnoMark (page 683).

Mark\_tracking\_translator (page 483)

This translator chooses which marks Mark\_engraver should engrave.

Music types accepted: ad-hoc-mark-event (page 49), coda-mark-event (page 51), rehearsal-mark-event (page 56), section-label-event (page 57), and segno-mark-event (page 57),

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 485)

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 `Staff_collecting_engraver` (page 499).

Music types accepted: `tempo-change-event` (page 59),

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 647).

**Output\_property\_engraver** (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

**Paper\_column\_engraver** (page 490)

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 50), and `label-event` (page 53),

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 655), and `PaperColumn` (page 665).

`Parenthesis_engraver` (page 491)

Parenthesize objects whose `parenthesize` property is `#t`.

This engraver creates the following layout object(s): `Parentheses` (page 666).

`Repeat_acknowledge_engraver` (page 494)

This translator adds entries to `repeatCommands` for events generated by `\\repeat volta`.

Music types accepted: `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 return-count`

End a repeated section. `return-count` is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. `repeat-count` is the number of times to perform this 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 496)

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 575), and `ControlPolygon` (page 576).

`Spacing_engraver` (page 498)

Make a `SpacingSpanner` and do bookkeeping of shortest starting and playing notes.

Music types accepted: `spacing-section-event` (page 57),

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` (non-negative exact rational or `+inf.0`)

Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s): `SpacingSpanner` (page 693).

`Spanner_tracking_engraver` (page 499)

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 499)

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 500)

This engraver ensures that stanza numbers are neatly aligned.

`System_start_delimiter_engraver` (page 501)

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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`Text_mark_engraver` (page 503)

Engraves arbitrary textual marks.

Music types accepted: `text-mark-event` (page 59),

Properties (read)

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

This engraver creates the following layout object(s): `TextMark` (page 719).

`Timing_translator` (page 505)

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 49), `bar-check-event` (page 50), `bar-event` (page 50), `fine-event` (page 52), and `partial-event` (page 55),

## 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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`timeSignatureFraction` (simple time signature)

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 first volta number for the current `\alternative` element. Not set outside of alternatives.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`timeSignatureFraction` (simple time signature)

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

`Tweak_engraver` (page 507)

Read the `tweaks` property from the originating event, and set properties.

Vertical\_align\_engraver (page 507)

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 698), and VerticalAlignment (page 742).

Volta\_engraver (page 507)

Make volta brackets.

Music types accepted: dal-segno-event (page 51), fine-event (page 52), and volta-span-event (page 60),

Properties (read)

currentCommandColumn (graphical (layout) object)

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

printTrivialVoltaRepeats (boolean)

Notate volta-style repeats even when the repeat count is 1.

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 *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

start-repeat *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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.

This engraver creates the following layout object(s): VoltaBracket (page 745), and VoltaBracketSpanner (page 747).

### 2.1.41 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 309).

This context creates the following layout object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), BarLine (page 538), BassFigure (page 544), BassFigureAlignment (page 544), BassFigureAlignmentPositioning (page 545), BassFigureBracket (page 546), BassFigureContinuation (page 547), BassFigureLine (page 548), Clef (page 565), ClefModifier (page 568), CueClef (page 577), CueEndClef (page 580), Custos (page 583), Divisio (page 585), DotColumn (page 588), FingeringColumn (page 605), InstrumentName (page 618), KeyCancellation (page 622), KeySignature (page 625), LedgerLineSpanner (page 631), NoteCollision (page 656), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PianoPedalBracket (page 672), RestCollision (page 679), ScriptColumn (page 680), ScriptRow (page 681), SostenuitoPedal (page 690), SostenuitoPedalLineSpanner (page 692), StaffEllipsis (page 696), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), SustainPedal (page 710), SustainPedalLineSpanner (page 711), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), and VerticalAxisGroup (page 742).

This context sets the following properties:

- Set context property alterationGlyphs to:
 

```
'((-1/2 . "accidentals.vaticanaM1")
 (0 . "accidentals.vaticana0")
 (1/2 . "accidentals.mensural1"))
```
- Set context property autoAccidentals to:
 

```
'(Staff #<procedure at lily/music-functions.scm:1714:0 (context pitch barnum)>)
```
- 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 `printTrivialVoltaRepeats` to `#t`.
- 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 557), to `item::extra-spacing-height-including-staff`.
- Set grob property `extra-spacing-width` in `BreathingSign` (page 557), to :  

```
'(-1.0 . 0.0)
```
- Set grob property `font-size` in `BreathingSign` (page 557), to -2.
- Set grob property `font-size` in `Divisio` (page 585), to -2.
- Set grob property `hair-thickness` in `BarLine` (page 538), to 0.65.
- Set grob property `ledger-line-thickness` in `StaffSymbol` (page 701), to :  

```
'(1 . 0)
```
- Set grob property `length-fraction` in `LedgerLineSpanner` (page 631), to 0.9.
- Set grob property `line-count` in `StaffSymbol` (page 701), to 4.
- Set grob property `neutral-direction` in `Custos` (page 583), to -1.
- Set grob property `neutral-position` in `Custos` (page 583), to 3.
- Set grob property `space-alist.clef` in `LeftEdge` (page 632), to :  

```
'(extra-space . 0)
```

- Set grob property `space-alist.custos` in `BarLine` (page 538), to :  
'(minimum-space . 0.7)
- Set grob property `space-alist.first-note` in `Clef` (page 565), to :  
'(minimum-fixed-space . 1.4)
- Set grob property `space-alist.right-edge` in `Custos` (page 583), to :  
'(extra-space . 0)
- Set grob property `style` in `Custos` (page 583), to 'vaticana.
- Set grob property `style` in `Dots` (page 588), to 'vaticana.
- Set grob property `thick-thickness` in `BarLine` (page 538), to 1.8.
- Set grob property `thickness` in `BreathingSign` (page 557), to 1.3.
- Set grob property `thickness` in `Divisio` (page 585), to 1.3.
- Set grob property `thickness` in `StaffSymbol` (page 701), to 0.5.

This is not a ‘Bottom’ context; search for such a one will commence after creating an implicit context of type `VaticanaVoice` (page 429).

Context `VaticanaStaff` can contain `CueVoice` (page 100), `NullVoice` (page 247), and `VaticanaVoice` (page 429).

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

`Accidental_engraver` (page 450)

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 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), and `AccidentalSuggestion` (page 529).

`Alteration_glyph_engraver` (page 451)

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 453)

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 742).

`Bar_engraver` (page 453)

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 49), `caesura-event` (page 51), `coda-mark-event` (page 51), `dal-segno-event` (page 51), `fine-event` (page 52), `section-event` (page 57), and `segno-mark-event` (page 57),

## 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 #t, `Bar_engraver` forbids line breaks where there is no bar line.

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

`printInitialRepeatBar` (boolean)  
 Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)  
 Notate volta-style repeats even when the repeat count is 1.

`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` *return-count*  
 End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*  
 Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Clef_engraver` (page 462)

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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Collision_engraver` (page 463)

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 656).

`Cue_clef_engraver` (page 465)

Determine and set reference point for pitches in cued voices.

Properties (read)

`clefTransposition` (integer)

Add this much extra transposition to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Custos_engraver` (page 466)

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 583).

`Divisio_engraver` (page 467)

Create divisions: chant notation for points of breathing or caesura.

Music types accepted: `caesura-event` (page 51), `fine-event` (page 52), `section-event` (page 57), `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 585).

`Dot_column_engraver` (page 467)

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 588).

`Figured_bass_engraver` (page 471)

Make figured bass numbers.

Music types accepted: `bass-figure-event` (page 50), and `rest-event` (page 56),

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 544), `BassFigureAlignment` (page 544), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), and `BassFigureLine` (page 548).

`Figured_bass_position_engraver` (page 472)

Position figured bass alignments over notes.

This engraver creates the following layout object(s):

`BassFigureAlignmentPositioning` (page 545).

`Fingering_column_engraver` (page 472)

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 605).

`Font_size_engraver` (page 472)

Put `fontSize` into font-size grob property.

Properties (read)

`fontSize` (number)

The relative size of all grobs in a context.

`Grob_pq_engraver` (page 476)

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.).

`Horizontal_script_engraver` (page 477)

Aligns Script horizontally

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.

`Instrument_name_engraver` (page 477)

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 618).

`Key_engraver` (page 479)

Engrave a key signature.

Music types accepted: `key-change-event` (page 53),

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 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 622), and `KeySignature` (page 625).

`Ledger_line_engraver` (page 481)

Create the spanner to draw ledger lines, and notices objects that need ledger lines.

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

`Merge_mmrest_numbers_engraver` (page 485)

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 488)

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 680).

`Optional_material_bracket_engraver` (page 489)

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Ottava_spanner_engraver` (page 489)

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

## 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 664).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Piano_pedal_align_engraver` (page 492)

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 692), `SustainPedalLineSpanner` (page 711), and `UnaCordaPedalLineSpanner` (page 740).

`Piano_pedal_engraver` (page 492)

Engrave piano pedal symbols and brackets.

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

## 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 672), SostenutoPedal (page 690), SustainPedal (page 710), and UnaCordaPedal (page 739).

Pure\_from\_neighbor\_engraver (page 494)

Coordinates items that get their pure heights from their neighbors.

Rest\_collision\_engraver (page 495)

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 679).

Script\_row\_engraver (page 496)

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Separating\_line\_group\_engraver (page 496)

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 700).

Skip\_typesetting\_engraver (page 497)

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 696).

Staff\_collecting\_engraver (page 499)

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 499)

Highlights music passages.

Music types accepted: staff-highlight-event (page 58),

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 699).

Staff\_symbol\_engraver (page 500)

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): StaffSymbol (page 701).

### 2.1.42 VaticanaVoice

A kind of Voice for typesetting Gregorian chant in a notational style approximating *Editio Vaticana*.

This context also accepts commands for the following context(s): Voice (page 439).

This context creates the following layout object(s): Arpeggio (page 535), Beam (page 549), BendAfter (page 551), BreathingSign (page 557), ClusterSpanner (page 569), ClusterSpannerBeacon (page 570), CombineTextScript (page 572), DotColumn (page 588), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicLineSpanner (page 595), DynamicText (page 597), DynamicTextSpanner (page 599), Episema (page 600), FingerGlideSpanner (page 601), Fingering (page 603), Glissando (page 610), Hairpin (page 614), InstrumentSwitch (page 619), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NoteColumn (page 657), NoteHead (page 658), NoteSpacing (page 660), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), Script (page 679), ScriptColumn (page 680), StringNumber (page 706), StrokeFinger (page 708), TextScript (page 721), Tie (page 725), TieColumn (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), VaticanaLigature (page 741), and VoiceFollower (page 744).

This context sets the following properties:

- Set context property autoBeaming to #f.
- Set grob property bound-details.left.padding in Episema (page 600), to 0.05.
- Set grob property bound-details.right.padding in Episema (page 600), to 0.05.
- Set grob property style in NoteHead (page 658), to 'vaticana.punctum.
- Set grob property thickness in Episema (page 600), to 2.5.

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 452)

Generate an Arpeggio symbol.

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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto\_beam\_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem\_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam\_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Bend\_engraver (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

Breathing\_sign\_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

Chord\_tremolo\_engraver (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster\_spanner\_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots\_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double\_percent\_repeat\_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.



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 590), and DoublePercentRepeatCounter (page 591).

Dynamic\_align\_engraver (page 469)

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 595).

Dynamic\_engraver (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 49), break-dynamic-span-event (page 50), and span-dynamic-event (page 57),

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 597), DynamicTextSpanner (page 599), and Hairpin (page 614).

Episema\_engraver (page 471)

Create an *Editio Vaticana*-style episema line.

Music types accepted: episema-event (page 52),

This engraver creates the following layout object(s): Episema (page 600).

**Finger\_glide\_engraver** (page 472)

Engraver to print a line between two *Fingering*, *StringNumber* or *StrokeFinger* grobs.

Music types accepted: *note-event* (page 55),

This engraver creates the following layout object(s): *FingerGlideSpanner* (page 601).

**Fingering\_engraver** (page 472)

Create fingering scripts.

Music types accepted: *fingering-event* (page 52),

This engraver creates the following layout object(s): *Fingering* (page 603).

**Font\_size\_engraver** (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: *glissando-event* (page 52),

Properties (read)

*glissandoMap* (list)

A map in the form of '((source1 . target1) (source2 . target2) (source<sub>n</sub> . target<sub>n</sub>))' 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 610).

**Grace\_auto\_beam\_engraver** (page 474)

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 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_beam\_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_engraver (page 475)

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 476)

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 478)

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 619).

`Laissez_vibrer_engraver` (page 481)

Create laissez vibrer items.

Music types accepted: `laissez-vibrer-event` (page 53),

This engraver creates the following layout object(s): `LaissezVibrerTie` (page 629), and `LaissezVibrerTieColumn` (page 630).

`Multi_measure_rest_engraver` (page 487)

Engrave multi-measure rests that are produced with ‘R’. It reads `measureStartNow` and `internalBarNumber` to determine what number to print over the `MultiMeasureRest` (page 649).

Music types accepted: `multi-measure-articulation-event` (page 54), `multi-measure-rest-event` (page 54), and `multi-measure-text-event` (page 54),

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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

`New_fingering_engraver` (page 487)

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 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

**Note\_head\_line\_engraver** (page 488)

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 744).

**Note\_heads\_engraver** (page 488)

Generate note heads.

Music types accepted: note-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.

staffLineLayoutFunction (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): NoteHead (page 658).

**Note\_spacing\_engraver** (page 489)

Generate NoteSpacing, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): NoteSpacing (page 660).

**Output\_property\_engraver** (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: apply-output-event (page 49),

**Part\_combine\_engraver** (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: note-event (page 55), and part-combine-event (page 55),

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 572).

**Percent\_repeat\_engraver** (page 492)

Make whole measure repeats.

Music types accepted: percent-event (page 56),

## 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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

`Repeat_tie_engraver` (page 494)

Create repeat ties.

Music types accepted: `repeat-tie-event` (page 56),

This engraver creates the following layout object(s): `RepeatTie` (page 676), and `RepeatTieColumn` (page 677).

`Rest_engraver` (page 495)

Engrave rests.

Music types accepted: `rest-event` (page 56),

## 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 678).

`Rhythmic_column_engraver` (page 495)

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): `NoteColumn` (page 657).

`Script_column_engraver` (page 495)

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 680).

`Script_engraver` (page 496)

Handle note scripted articulations.

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

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 679).

Slash\_repeat\_engraver (page 497)

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

Spanner\_break\_forbid\_engraver (page 499)

Forbid breaks in certain spanners.

Text\_engraver (page 503)

Create text scripts.

Music types accepted: text-script-event (page 59),

This engraver creates the following layout object(s): TextScript (page 721).

Tie\_engraver (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 59),

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 725), and TieColumn (page 727).

Trill\_spanner\_engraver (page 506)

Create trill spanners.

Music types accepted: trill-span-event (page 60),

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 734).

`Tuplet_engraver` (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

`Vaticana_ligature_engraver` (page 507)

Handle ligatures by glueing special ligature heads together.

Music types accepted: `ligature-event` (page 53), and `pes-or-flexa-event` (page 56),

This engraver creates the following layout object(s): `DotColumn` (page 588), and `VaticanaLigature` (page 741).

### 2.1.43 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 535), `Beam` (page 549), `BendAfter` (page 551), `BreathingSign` (page 557), `ClusterSpanner` (page 569), `ClusterSpannerBeacon` (page 570), `CombineTextScript` (page 572), `Dots` (page 588), `DoublePercentRepeat` (page 590), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `FingerGlideSpanner` (page 601), `Fingering` (page 603), `Flag` (page 605), `Glissando` (page 610), `Hairpin` (page 614), `InstrumentSwitch` (page 619), `LaissezVibrerTie` (page 629), `LaissezVibrerTieColumn` (page 630), `LigatureBracket` (page 634), `MultiMeasureRest` (page 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), `MultiMeasureRestText` (page 653), `NoteColumn` (page 657), `NoteHead` (page 658), `NoteSpacing` (page 660), `PercentRepeat` (page 667), `PercentRepeatCounter` (page 668), `PhrasingSlur` (page 670), `RepeatSlash` (page 675), `RepeatTie` (page 676), `RepeatTieColumn` (page 677), `Rest` (page 678), `Script` (page 679), `ScriptColumn` (page 680), `Slur` (page 688), `Stem` (page 703), `StemStub` (page 705), `StemTremolo` (page 705), `StringNumber` (page 706), `StrokeFinger` (page 708), `TextScript` (page 721), `TextSpanner` (page 723), `Tie` (page 725), `TieColumn` (page 727), `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), `TrillPitchParentheses` (page 733), `TrillSpanner` (page 734), `TupletBracket` (page 736), `TupletNumber` (page 737), and `VoiceFollower` (page 744).

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 452)

Generate an Arpeggio symbol.



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

This engraver creates the following layout object(s): Arpeggio (page 535).

Auto\_beam\_engraver (page 452)

Generate beams based on measure characteristics and observed Stems. Uses beatBase, beatStructure, beamExceptions, measureLength, and measurePosition to decide when to start and stop a beam. Overriding beaming is done through Stem\_engraver (page 500), properties stemLeftBeamCount and stemRightBeamCount.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

autoBeaming (boolean)

If set to #t then beams are generated automatically. If set to #f, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

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.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Beam\_engraver (page 457)

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

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Bend\_engraver (page 459)

Create fall spanners.

Music types accepted: bend-after-event (page 50),

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 551).

Breathing\_sign\_engraver (page 460)

Notate breath marks.

Music types accepted: breathing-event (page 51),

Properties (read)

breathMarkType (symbol)

The type of BreathingSign to create at \breathe.

This engraver creates the following layout object(s): BreathingSign (page 557).

Chord\_tremolo\_engraver (page 462)

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Cluster\_spanner\_engraver (page 463)

Engrave a cluster using Spanner notation.

Music types accepted: cluster-note-event (page 51),

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

Dots\_engraver (page 468)

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Double\_percent\_repeat\_engraver (page 468)

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Dynamic\_align\_engraver (page 469)

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 595).

Dynamic\_engraver (page 470)

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: absolute-dynamic-event (page 49), break-dynamic-span-event (page 50), and span-dynamic-event (page 57),

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 597), DynamicTextSpanner (page 599), and Hairpin (page 614).

Finger\_glide\_engraver (page 472)

Engraver to print a line between two Fingering, StringNumber or StrokeFinger grobs.

Music types accepted: note-event (page 55),

This engraver creates the following layout object(s): FingerGlideSpanner (page 601).

Fingering\_engraver (page 472)

Create fingering scripts.

Music types accepted: `fingering-event` (page 52),

This engraver creates the following layout object(s): `Fingering` (page 603).

Font\_size\_engraver (page 472)

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 473)

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 474)

Engrave glissandi.

Music types accepted: `glissando-event` (page 52),

Properties (read)

`glissandoMap` (list)

A map in the form of `'((source1 . target1) (source2 . target2) (source . 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 610).

Grace\_auto\_beam\_engraver (page 474)

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 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, autobeaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): `Beam` (page 549).

Grace\_beam\_engraver (page 475)

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 50),

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_engraver (page 475)

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 476)

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 478)

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 619).

Laissez\_vibrer\_engraver (page 481)

Create laissez vibrer items.

Music types accepted: laissez-vibrer-event (page 53),

This engraver creates the following layout object(s): LaissezVibrerTie (page 629), and LaissezVibrerTieColumn (page 630).

**Ligature\_bracket\_engraver** (page 481)

Handle Ligature\_events by engraving Ligature brackets.

Music types accepted: `ligature-event` (page 53),

This engraver creates the following layout object(s): `LigatureBracket` (page 634).

**Multi\_measure\_rest\_engraver** (page 487)

Engrave multi-measure rests that are produced with ‘R’. It reads `measureStartNow` and `internalBarNumber` to determine what number to print over the `MultiMeasureRest` (page 649).

Music types accepted: `multi-measure-articulation-event` (page 54),

`multi-measure-rest-event` (page 54), and `multi-measure-text-event` (page 54),

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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

**New\_fingering\_engraver** (page 487)

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 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

**Note\_head\_line\_engraver** (page 488)

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 744).

`Note_heads_engraver` (page 488)

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_spacing_engraver` (page 489)

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Output_property_engraver` (page 490)

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Part_combine_engraver` (page 491)

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

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 572).

`Percent_repeat_engraver` (page 492)

Make whole measure repeats.

Music types accepted: `percent-event` (page 56),

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 667), and `PercentRepeatCounter` (page 668).

`Phrasing_slur_engraver` (page 492)

Print phrasing slurs. Similar to `Slur_engraver` (page 497).

Music types accepted: `note-event` (page 55), and `phrasing-slur-event` (page 56),

This engraver creates the following layout object(s): `PhrasingSlur` (page 670).

`Pitched_trill_engraver` (page 494)

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): `TrillPitchAccidental` (page 730), `TrillPitchGroup` (page 731), `TrillPitchHead` (page 733), and `TrillPitchParentheses` (page 733).

`Repeat_tie_engraver` (page 494)

Create repeat ties.

Music types accepted: `repeat-tie-event` (page 56),

This engraver creates the following layout object(s): `RepeatTie` (page 676), and `RepeatTieColumn` (page 677).

`Rest_engraver` (page 495)

Engrave rests.

Music types accepted: `rest-event` (page 56),

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 678).

`Rhythmic_column_engraver` (page 495)

Generate `NoteColumn`, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): `NoteColumn` (page 657).

`Script_column_engraver` (page 495)

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 680).

`Script_engraver` (page 496)

Handle note scripted articulations.

Music types accepted: `articulation-event` (page 50),

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 679).

`Slash_repeat_engraver` (page 497)

Make beat repeats.

Music types accepted: `repeat-slash-event` (page 56),

This engraver creates the following layout object(s): `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

`Slur_engraver` (page 497)

Build slur grobs from slur events.

Music types accepted: `note-event` (page 55), and `slur-event` (page 57),

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 688).

`Spanner_break_forbid_engraver` (page 499)

Forbid breaks in certain spanners.

`Stem_engraver` (page 500)

Create stems, flags and single-stem tremolos. It also works together with the beam engraver for overriding beaming.

Music types accepted: `tremolo-event` (page 59),

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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Text_engraver` (page 503)

Create text scripts.

Music types accepted: `text-script-event` (page 59),

This engraver creates the following layout object(s): `TextScript` (page 721).

`Text_spanner_engraver` (page 503)

Create text spanner from an event.

Music types accepted: `text-span-event` (page 59),

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 723).

`Tie_engraver` (page 503)

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

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 725), and `TieColumn` (page 727).

`Trill_spanner_engraver` (page 506)

Create trill spanners.

Music types accepted: `trill-span-event` (page 60),

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 734).

`Tuplet_engraver` (page 506)

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

## 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 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), and `AccidentalSuggestion` (page 529).

`Accidental_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), and `VaticanaStaff` (page 414).

### 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 numbers, 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 69), `ChordNames` (page 98), `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `NoteNames` (page 245), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.3 Ambitus\_engraver

Create an ambitus.

Properties (read)

`keyAlterations` (list)

The current key signature. This is an alist containing `(step . alter)` or `((octave . step) . alter)`, where `step` is a number in the range 0 to 6 and `alter` a fraction, denoting alteration. For alterations, use symbols, e.g., `keyAlterations = #`((6 . ,FLAT))`.

`middleCClefPosition` (number)

The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`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 528), `Ambitus` (page 531), `AmbitusAccidental` (page 533), `AmbitusLine` (page 533), and `AmbitusNoteHead` (page 534).

`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 535).

`Arpeggio_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.5 `Auto_beam_engraver`

Generate beams based on measure characteristics and observed Stems. Uses `beatBase`, `beatStructure`, `beamExceptions`, `measureLength`, and `measurePosition` to decide when to start and stop a beam. Overriding beaming is done through `Stem_engraver` (page 500), properties `stemLeftBeamCount` and `stemRightBeamCount`.

Music types accepted: `beam-forbid-event` (page 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

`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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): Beam (page 549).

Auto\_beam\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 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 742).

Axis\_group\_engraver is part of the following context(s) in \layout: ChordGrid (page 69), ChordNames (page 98), DrumStaff (page 111), Dynamics (page 130), FiguredBass (page 135), FretBoards (page 136), GregorianTranscriptionLyrics (page 141), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), Lyrics (page 218), MensuralStaff (page 221), NoteNames (page 245), OneStaff (page 249), PetrucciStaff (page 250), RhythmicStaff (page 277), Staff (page 309), StandaloneRhythmStaff (page 349), TabStaff (page 365), VaticanaLyrics (page 388), and VaticanaStaff (page 414).

### 2.2.7 Balloon\_engraver

Create balloon texts.

Music types accepted: annotate-output-event (page 49),

This engraver creates the following layout object(s): BalloonText (page 537).

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 49), caesura-event (page 51), coda-mark-event (page 51), dal-segno-event (page 51), fine-event (page 52), section-event (page 57), and segno-mark-event (page 57),

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 `#t`, `Bar_engraver` forbids line breaks where there is no bar line.

`measureBarType` (string)

Bar line to insert at a measure boundary.

`printInitialRepeatBar` (boolean)

Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`printTrivialVoltaRepeats` (boolean)

Notate volta-style repeats even when the repeat count is 1.

`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 return-count`

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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 538).

`Bar_engraver` is part of the following context(s) in `\layout`: `ChordGrid` (page 69), `DrumStaff` (page 111), `Dynamics` (page 130), `GregorianTranscriptionStaff` (page 144),



InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), RhythmicStaff (page 277), Staff (page 309), StandaloneRhythmStaff (page 349), TabStaff (page 365), and VaticanaStaff (page 414).

### 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 `Staff_collecting_engraver` (page 499). 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 first volta number for the current \alternative element. Not set outside of alternatives.

`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)

The current point within the measure.

`stavesFound` (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): `BarNumber` (page 542), and `CenteredBarNumber` (page 561).

`Bar_number_engraver` is part of the following context(s) in `\layout`: `Score` (page 283), and `StandaloneRhythmScore` (page 323).

### 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 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 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 50),

Properties (read)

`beamMelismaBusy` (boolean)

Signal if a beam is present.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`subdivideBeams` (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beam-lets, dependent on `beamMaximumSubdivision`, between beats at multiples of `beamMinimumSubdivision`.

This engraver creates the following layout object(s): `Beam` (page 549).

`Beam_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice`

(page 207), MensuralVoice (page 235), NullVoice (page 247), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.12 Beam\_performer

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

Beam\_performer is part of the following context(s) in \midi: ChordNames (page 98), CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), NullVoice (page 247), PetrucciVoice (page 265), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.13 Beat\_engraver

This engraver is just a functionally identical copy of Section 2.2.14 [Beat\_performer], page 458, used for visualising its effects. You can also use it for showcasing the effects of the current beatStructure.

Music types accepted: articulation-event (page 50), and note-event (page 55),

Properties (read)

barExtraVelocity (integer)

Extra MIDI velocity added by the ‘Beat\_performer’ at the start of each measure.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatExtraVelocity (integer)

Extra MIDI velocity added by the ‘Beat\_performer’ at the start of each beat.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

measurePosition (moment)

The current point within the measure.

timeSignatureFraction (simple time signature)

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 50), and note-event (page 55),

Properties (read)

barExtraVelocity (integer)

Extra MIDI velocity added by the ‘Beat\_performer’ at the start of each measure.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatExtraVelocity (integer)

Extra MIDI velocity added by the ‘Beat\_performer’ at the start of each beat.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

measurePosition (moment)

The current point within the measure.

timeSignatureFraction (simple time signature)

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 50),

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 551).

Bend\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.16 Bend\_spanner\_engraver

Engraver to print a BendSpanner.

Music types accepted: bend-span-event (page 50), note-event (page 55), and string-number-event (page 58),

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 552).

`Bend_spanner_engraver` is part of the following context(s) in `\layout`: `TabVoice` (page 377).

### 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 554), `BreakAlignment` (page 555), and `LeftEdge` (page 632).

`Break_align_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 2.2.18 `Breathing_sign_engraver`

Notate breath marks.

Music types accepted: `breathing-event` (page 51),

Properties (read)

`breathMarkType` (symbol)

The type of `BreathingSign` to create at `\breathe`.

This engraver creates the following layout object(s): `BreathingSign` (page 557).

`Breathing_sign_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 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 51),

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 557), and `CaesuraScript` (page 559).

`Caesura_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `StandaloneRhythmStaff` (page 349), and `TabStaff` (page 365).

### 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 562).

`Centered_bar_number_align_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 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 563).

Chord\_name\_engraver is part of the following context(s) in \layout: ChordNames (page 98).

**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 564).

Chord\_square\_engraver is part of the following context(s) in \layout: ChordGrid (page 69).

**2.2.23 Chord\_tremolo\_engraver**

Generate beams for tremolo repeats.

Music types accepted: tremolo-span-event (page 59),

This engraver creates the following layout object(s): Beam (page 549).

Chord\_tremolo\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

**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 to a clef. Values of 7 and -7 are common.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef 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 565), and `ClefModifier` (page 568).

`Clef_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.25 Cluster\_spanner\_engraver

Engrave a cluster using `Spanner` notation.

Music types accepted: `cluster-note-event` (page 51),

This engraver creates the following layout object(s): `ClusterSpanner` (page 569), and `ClusterSpannerBeacon` (page 570).

`Cluster_spanner_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 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 656).

`Collision_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 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 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` (positive exact rational or `+inf.0`)

Sub-bar unit of completion.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`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 658), `Tie` (page 725), and `TieColumn` (page 727).

`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 56),

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` (positive exact rational or `+inf.0`)

Sub-bar unit of completion.

`measureLength` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`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 678).

`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 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 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`: `ChordGridScore` (page 74), and `Score` (page 283).

### 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 to a clef. 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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 568), `CueClef` (page 577), and `CueEndClef` (page 580).

`Cue_clef_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 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 52), and `note-event` (page 55),

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 69), and `ChordNames` (page 98).

### 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 583).

`Custos_engraver` is part of the following context(s) in `\layout`: `InternalMensuralStaff` (page 180), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), and `VaticanaStaff` (page 414).

### 2.2.34 `Divisio_engraver`

Create divisiones: chant notation for points of breathing or caesura.

Music types accepted: `caesura-event` (page 51), `fine-event` (page 52), `section-event` (page 57), `volta-repeat-end-event` (page 60), and `volta-repeat-start-event` (page 60),

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 585).

`Divisio_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), and `VaticanaStaff` (page 414).

### 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 588).

`Dot_column_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff`

(page 221), PetrucciStaff (page 250), RhythmicStaff (page 277), Staff (page 309), StandaloneRhythmStaff (page 349), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.36 Dots\_engraver

Create Dots (page 588), objects for rhythmic-head-interface (page 813)s.

This engraver creates the following layout object(s): Dots (page 588).

Dots\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.37 Double\_percent\_repeat\_engraver

Make double measure repeats.

Music types accepted: double-percent-event (page 51),

Properties (read)

countPercentRepeats (boolean)

If set, produce counters for percent repeats.

measureLength (positive exact rational or +inf.0)

The musical length of the current measure.

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 590), and DoublePercentRepeatCounter (page 591).

Double\_percent\_repeat\_engraver is part of the following context(s) in \layout: ChordGrid (page 69), CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.38 Drum\_note\_performer

Play drum notes.

Music types accepted: articulation-event (page 50), note-event (page 55), and tie-event (page 59),

Drum\_note\_performer is part of the following context(s) in \midi: DrumVoice (page 120).

### 2.2.39 Drum\_notes\_engraver

Generate drum note heads.

Music types accepted: note-event (page 55),

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 658), and `Script` (page 679).

`Drum_notes_engraver` is part of the following context(s) in `\layout`: `DrumVoice` (page 120).

### 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 52),

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.).

`endAtSkip` (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 593).

`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 595).

`Dynamic_align_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `Dynamics` (page 130), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.42 `Dynamic_engraver`

Create hairpins, dynamic texts and dynamic text spanners.

Music types accepted: `absolute-dynamic-event` (page 49), `break-dynamic-span-event` (page 50), and `span-dynamic-event` (page 57),

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 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

`Dynamic_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `Dynamics` (page 130), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.43 `Dynamic_performer`

Music types accepted: `absolute-dynamic-event` (page 49), `crescendo-event` (page 51), and `decrescendo-event` (page 51),

Properties (read)

`dynamicAbsoluteVolumeFunction` (procedure)

A procedure that takes one argument, the text value of a dynamic event, and returns the absolute volume of that dynamic event.

`instrumentEqualizer` (procedure)

A function taking a string (instrument name), and returning a (*min* . *max*) pair of numbers for the loudness range of the instrument.

`midiInstrument` (string)

Name of the MIDI instrument to use.

`midiMaximumVolume` (number)

Analogous to `midiMinimumVolume`.

`midiMinimumVolume` (number)

Set the minimum loudness for MIDI. Ranges from 0 to 1.

`Dynamic_performer` is part of the following context(s) in `\midi`: `ChordNames` (page 98), `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.44 Episema\_engraver

Create an *Editio Vaticana*-style episema line.

Music types accepted: episema-event (page 52),

This engraver creates the following layout object(s): Episema (page 600).

Episema\_engraver is part of the following context(s) in \layout: GregorianTranscriptionVoice (page 157), and VaticanaVoice (page 429).

### 2.2.45 Extender\_engraver

Create lyric extenders.

Music types accepted: completize-extender-event (page 51), and extender-event (page 52),

Properties (read)

extendersOverRests (boolean)

Whether to continue extenders as they cross a rest.

This engraver creates the following layout object(s): LyricExtender (page 635).

Extender\_engraver is part of the following context(s) in \layout: GregorianTranscriptionLyrics (page 141), Lyrics (page 218), and VaticanaLyrics (page 388).

### 2.2.46 Figured\_bass\_engraver

Make figured bass numbers.

Music types accepted: bass-figure-event (page 50), and rest-event (page 56),

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 544), BassFigureAlignment (page 544), BassFigureBracket (page 546), BassFigureContinuation (page 547), and BassFigureLine (page 548).

Figured\_bass\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), FiguredBass (page 135), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).



### 2.2.47 Figured\_bass\_position\_engraver

Position figured bass alignments over notes.

This engraver creates the following layout object(s): BassFigureAlignmentPositioning (page 545).

Figured\_bass\_position\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.48 Finger\_glide\_engraver

Engraver to print a line between two Fingering, StringNumber or StrokeFinger grobs.

Music types accepted: note-event (page 55),

This engraver creates the following layout object(s): FingerGlideSpanner (page 601).

Finger\_glide\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 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 605).

Fingering\_column\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.50 Fingering\_engraver

Create fingering scripts.

Music types accepted: fingering-event (page 52),

This engraver creates the following layout object(s): Fingering (page 603).

Fingering\_engraver is part of the following context(s) in \layout: CueVoice (page 100), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), VaticanaVoice (page 429), and Voice (page 439).

### 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 100), DrumStaff (page 111), DrumVoice (page 120), Dynamics (page 130), FretBoards (page 136), GregorianTranscriptionLyrics (page 141), GregorianTranscriptionStaff (page 144), GregorianTranscriptionVoice (page 157), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), KievanVoice (page 207),

Lyrics (page 218), MensuralStaff (page 221), MensuralVoice (page 235), PetrucciStaff (page 250), PetrucciVoice (page 265), RhythmicStaff (page 277), Staff (page 309), StandaloneRhythmStaff (page 349), StandaloneRhythmVoice (page 354), TabStaff (page 365), TabVoice (page 377), VaticanaLyrics (page 388), VaticanaStaff (page 414), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.52 Footnote\_engraver

Create footnote texts.

This engraver creates the following layout object(s): Footnote (page 606).

Footnote\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 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 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.54 Fretboard\_engraver

Generate fret diagram from one or more events of type NoteEvent.

Music types accepted: fingering-event (page 52), note-event (page 55), and string-number-event (page 58),

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 608).

Fretboard\_engraver is part of the following context(s) in `\layout`: FretBoards (page 136).

### 2.2.55 Glissando\_engraver

Engrave glissandi.

Music types accepted: glissando-event (page 52),

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 610).

Glissando\_engraver is part of the following context(s) in `\layout`: CueVoice (page 100), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 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 autobeaming, just like setting the context property `'autoBeaming'` to `##f`.

Music types accepted: beam-forbid-event (page 50),

Properties (read)

`autoBeaming` (boolean)

If set to `#t` then beams are generated automatically. If set to `#f`, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_auto\_beam\_engraver is part of the following context(s) in `\layout`: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.57 Grace\_beam\_engraver

Handle Beam events by engraving beams. If omitted, then notes are printed with flags instead of beams. Only engraves beams when we are at grace points in time.

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

Properties (read)

beamMelismaBusy (boolean)

Signal if a beam is present.

beatBase (positive exact rational or +inf.0)

The musical length corresponding to one unit of beatStructure.

beatStructure (list)

A sequence describing the length of each beat in the measure in units of beatBase.

subdivideBeams (boolean)

If set, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

This engraver creates the following layout object(s): Beam (page 549).

Grace\_beam\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 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 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 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 611).

Grace\_spacing\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 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 612).

`Grid_chord_name_engraver` is part of the following context(s) in `\layout`: `ChordGrid` (page 69).

### 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 612).

`Grid_line_span_engraver` is not part of any context

### 2.2.62 Grid\_point\_engraver

Generate grid points.

Properties (read)

`gridInterval` (positive exact rational or `+inf.0`)

Interval for which to generate `GridPoints`.

This engraver creates the following layout object(s): `GridPoint` (page 613).

`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 (*end-moment* . *grob*) cons cells. This is for internal (C++) use only.

This property contains the grobs which are still busy (e.g., note heads, spanners, etc.).

Properties (write)

`busyGrobs` (list)

A queue of (*end-moment* . *grob*) cons cells. This is for internal (C++) use only.

This property contains the grobs which are still busy (e.g., note heads, spanners, etc.).

`Grob_pq_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumStaff` (page 111), `DrumVoice` (page 120), `GregorianTranscriptionStaff` (page 144), `GregorianTranscriptionVoice` (page 157), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `KievanVoice` (page 207), `MensuralStaff` (page 221), `MensuralVoice` (page 235), `NullVoice`

(page 247), PetrucciStaff (page 250), PetrucciVoice (page 265), Staff (page 309), StandaloneRhythmVoice (page 354), TabStaff (page 365), TabVoice (page 377), VaticanaStaff (page 414), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.64 Horizontal\_bracket\_engraver

Create horizontal brackets over notes for musical analysis purposes.

Music types accepted: note-grouping-event (page 55),

This engraver creates the following layout object(s): HorizontalBracket (page 615), and HorizontalBracketText (page 617).

Horizontal\_bracket\_engraver is not part of any context

### 2.2.65 Horizontal\_script\_engraver

Aligns Script horizontally

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.

Horizontal\_script\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.66 Hyphen\_engraver

Create lyric hyphens, vowel transitions and distance constraints between words.

Music types accepted: hyphen-event (page 53), and vowel-transition-event (page 60),

This engraver creates the following layout object(s): LyricHyphen (page 636), LyricSpace (page 639), and VowelTransition (page 748).

Hyphen\_engraver is part of the following context(s) in \layout: GregorianTranscriptionLyrics (page 141), Lyrics (page 218), and VaticanaLyrics (page 388).

### 2.2.67 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 618).

`Instrument_name_engraver` is part of the following context(s) in `\layout`: `ChoirStaff` (page 67), `DrumStaff` (page 111), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Staff` (page 309), `StaffGroup` (page 322), `StandaloneRhythmStaff` (page 349), `TabStaff` (page 365), `VaticanaLyrics` (page 388), and `VaticanaStaff` (page 414).

## 2.2.68 `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 619).

`Instrument_switch_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

## 2.2.69 `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 499), also needs to be there so that marks appear at the intended Y location.

Music types accepted: `ad-hoc-jump-event` (page 49), `dal-segno-event` (page 51), and `fine-event` (page 52),

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 621).

*Jump\_engraver* is part of the following context(s) in \layout: *ChordGridScore* (page 74), *Score* (page 283), *StandaloneRhythmScore* (page 323), and *VaticanaScore* (page 390).

### 2.2.70 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 275).

### 2.2.71 Key\_engraver

Engrave a key signature.

Music types accepted: *key-change-event* (page 53),

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 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 622), and `KeySignature` (page 625).

`Key_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), and `VaticanaStaff` (page 414).

### 2.2.72 `Key_performer`

Music types accepted: `key-change-event` (page 53),

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 111), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.73 Kievan\_ligature\_engraver

Handle Kievan\_ligature\_events by glueing Kievan heads together.

Music types accepted: `ligature-event` (page 53),

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

`Kievan_ligature_engraver` is part of the following context(s) in `\layout`: `KievanVoice` (page 207).

### 2.2.74 Laissez\_vibrer\_engraver

Create laissez vibrer items.

Music types accepted: `laissez-vibrer-event` (page 53),

This engraver creates the following layout object(s): `LaissezVibrerTie` (page 629), and `LaissezVibrerTieColumn` (page 630).

`Laissez_vibrer_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.75 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 631).

`Ledger_line_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `StandaloneRhythmStaff` (page 349), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.76 Ligature\_bracket\_engraver

Handle Ligature\_events by engraving Ligature brackets.

Music types accepted: `ligature-event` (page 53),

This engraver creates the following layout object(s): `LigatureBracket` (page 634).

`Ligature_bracket_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), and `Voice` (page 439).

### 2.2.77 Lyric\_engraver

Engrave text for lyrics.

Music types accepted: `lyric-event` (page 53),

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 640).

`Lyric_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionLyrics` (page 141), `Lyrics` (page 218), and `VaticanaLyrics` (page 388).

### 2.2.78 Lyric\_performer

Music types accepted: lyric-event (page 53),

Lyric\_performer is part of the following context(s) in \midi:  
GregorianTranscriptionLyrics (page 141), and Lyrics (page 218).

### 2.2.79 Lyric\_repeat\_count\_engraver

Create repeat counts within lyrics for modern transcriptions of Gregorian chant.

Music types accepted: volta-repeat-end-event (page 60),

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 637).

Lyric\_repeat\_count\_engraver is part of the following context(s) in \layout:  
GregorianTranscriptionLyrics (page 141).

### 2.2.80 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 499), 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 483). 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 570), RehearsalMark (page 673), SectionLabel (page 681), and SegnoMark (page 683).

Mark\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.81 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: ChordGridScore (page 74), and Score (page 283).

### 2.2.82 Mark\_tracking\_translator

This translator chooses which marks Mark\_engraver should engrave.

Music types accepted: ad-hoc-mark-event (page 49), coda-mark-event (page 51), rehearsal-mark-event (page 56), section-label-event (page 57), and segno-mark-event (page 57),

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.

`Mark_tracking_translator` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390); in `\midi`: `ChordGridScore` (page 74), and `Score` (page 283).

### 2.2.83 `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: `measure-counter-event` (page 53),

Properties (read)

`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.).

`measurePosition` (moment)

The current point within the measure.

This engraver creates the following layout object(s): `MeasureCounter` (page 641).

`Measure_counter_engraver` is not part of any context

### 2.2.84 `Measure_grouping_engraver`

Create `MeasureGrouping` to indicate beat subdivision.

Properties (read)

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`beatStructure` (list)

A sequence describing the length of each beat in the measure in units of `beatBase`.

`currentMusicalColumn` (graphical (layout) object)

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

`measurePosition` (moment)

The current point within the measure.

This engraver creates the following layout object(s): `MeasureGrouping` (page 643).

`Measure_grouping_engraver` is not part of any context

### 2.2.85 `Measure_spanner_engraver`

This engraver creates spanners bounded by the columns that start and end measures in response to `\startMeasureSpanner` and `\stopMeasureSpanner`.

Music types accepted: `measure-spanner-event` (page 54),

Properties (read)

`currentCommandColumn` (graphical (layout) object)

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

`measurePosition` (moment)

The current point within the measure.

This engraver creates the following layout object(s): `MeasureSpanner` (page 644).

`Measure_spanner_engraver` is not part of any context

### 2.2.86 `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 `#t`.

This engraver creates the following layout object(s): `MelodyItem` (page 646).

`Melody_engraver` is not part of any context

### 2.2.87 `Mensural_ligature_engraver`

Handle `Mensural_ligature_events` by glueing special ligature heads together.

Music types accepted: `ligature-event` (page 53),

This engraver creates the following layout object(s): `MensuralLigature` (page 646).

`Mensural_ligature_engraver` is part of the following context(s) in `\layout`:  
`MensuralVoice` (page 235), and `PetrucchiVoice` (page 265).

### 2.2.88 `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 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.89 `Merge_rests_engraver`

Engraver to merge rests in multiple voices on the same staff. This works by gathering all rests at a time step. If they are all of the same length and there are at least two they are moved to the correct location as if there were one voice.

Properties (read)

`suspendRestMerging` (boolean)

When using the `Merge_rest_engraver` do not merge rests when this is set to `#t`.

`Merge_rests_engraver` is not part of any context

### 2.2.90 `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 `Staff_collecting_engraver` (page 499).

Music types accepted: tempo-change-event (page 59),

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 647).

`Metronome_mark_engraver` is part of the following context(s) in `\layout`:

`ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 2.2.91 `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 111), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.92 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 `MultiMeasureRest` (page 649).

Music types accepted: `multi-measure-articulation-event` (page 54), `multi-measure-rest-event` (page 54), and `multi-measure-text-event` (page 54),

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 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 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

`Multi_measure_rest_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.93 New\_fingering\_engraver

Create fingering scripts for notes in a new chord. This engraver is ill-named, since it also takes care of articulations and harmonic note heads.

Properties (read)

`fingeringOrientations` (list)

A list of symbols, containing ‘left’, ‘right’, ‘up’ and/or ‘down’. This list determines where fingerings are put relative to the chord being fingered.

`harmonicDots` (boolean)

If set, harmonic notes in dotted chords get dots.

`stringNumberOrientations` (list)

See `fingeringOrientations`.

`strokeFingerOrientations` (list)

See `fingeringOrientations`.

This engraver creates the following layout object(s): `Fingering` (page 603), `Script` (page 679), `StringNumber` (page 706), and `StrokeFinger` (page 708).

`New_fingering_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `VaticanaVoice` (page 429), and `Voice` (page 439).



### 2.2.94 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 680).

`Non_musical_script_column_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.95 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 744).

`Note_head_line_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.96 Note\_heads\_engraver

Generate note heads.

Music types accepted: `note-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`.

`staffLineLayoutFunction` (procedure)

Layout of staff lines, traditional, or semitone.

This engraver creates the following layout object(s): `NoteHead` (page 658).

`Note_heads_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `NullVoice` (page 247), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.97 Note\_name\_engraver

Print pitches as words.

Music types accepted: `note-event` (page 55),

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 659).

`Note_name_engraver` is part of the following context(s) in `\layout`: `NoteNames` (page 245).

### 2.2.98 Note\_performer

Music types accepted: `articulation-event` (page 50), `breathing-event` (page 51), `note-event` (page 55), and `tie-event` (page 59),

`Note_performer` is part of the following context(s) in `\midi`: `ChordNames` (page 98), `CueVoice` (page 100), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.99 Note\_spacing\_engraver

Generate `NoteSpacing`, an object linking horizontal lines for use in spacing.

This engraver creates the following layout object(s): `NoteSpacing` (page 660).

`Note_spacing_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.100 Optional\_material\_bracket\_engraver

Notate in-staff brackets for optional material.

Music types accepted: `optional-material-event` (page 55),

This engraver creates the following layout object(s): `OptionalMaterialBracket` (page 661).

`Optional_material_bracket_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.101 Ottava\_spanner\_engraver

Create a text spanner when the ottavation property changes.

Music types accepted: `ottava-event` (page 55),

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 664).

`Ottava_spanner_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), and `VaticanaStaff` (page 414).

### 2.2.102 `Output_property_engraver`

Apply a procedure to any grob acknowledged.

Music types accepted: `apply-output-event` (page 49),

`Output_property_engraver` is part of the following context(s) in `\layout`: `ChoirStaff` (page 67), `ChordGrid` (page 69), `ChordGridScore` (page 74), `ChordNames` (page 98), `CueVoice` (page 100), `DrumStaff` (page 111), `DrumVoice` (page 120), `Dynamics` (page 130), `FretBoards` (page 136), `GrandStaff` (page 139), `GregorianTranscriptionStaff` (page 144), `GregorianTranscriptionVoice` (page 157), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `KievanVoice` (page 207), `MensuralStaff` (page 221), `MensuralVoice` (page 235), `PetrucchiStaff` (page 250), `PetrucchiVoice` (page 265), `PianoStaff` (page 275), `RhythmicStaff` (page 277), `Score` (page 283), `Staff` (page 309), `StaffGroup` (page 322), `StandaloneRhythmScore` (page 323), `StandaloneRhythmStaff` (page 349), `StandaloneRhythmVoice` (page 354), `TabStaff` (page 365), `TabVoice` (page 377), `VaticanaScore` (page 390), `VaticanaStaff` (page 414), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.103 `Page_turn_engraver`

Decide where page turns are allowed to go.

Music types accepted: `break-event` (page 50),

Properties (read)

`currentBarLine` (graphical (layout) object)

Set to the `BarLine` that `Bar_engraver` has created in the current timestep.

`pageTurnMinimumRepeatLength` (non-negative exact rational or `+inf.0`)

Minimum length of a repeated section for a page turn to be allowed within that section.

`pageTurnMinimumRestLength` (non-negative exact rational or `+inf.0`)

Minimum length of a rest for a page turn to be allowed.

`Page_turn_engraver` is not part of any context

### 2.2.104 `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 50), and `label-event` (page 53),

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 655), and `PaperColumn` (page 665).

`Paper_column_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

**2.2.105 Parenthesis\_engraver**

Parenthesize objects whose `parenthesize` property is #t.

This engraver creates the following layout object(s): `Parentheses` (page 666).

`Parenthesis_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

**2.2.106 Part\_combine\_engraver**

Part combine engraver for orchestral scores: Print markings ‘a2’, ‘Solo’, ‘Solo II’, and ‘unisono’.

Music types accepted: `note-event` (page 55), and `part-combine-event` (page 55),

**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 572).

`Part_combine_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.107 Percent\_repeat\_engraver

Make whole measure repeats.

Music types accepted: percent-event (page 56),

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 667), and PercentRepeatCounter (page 668).

Percent\_repeat\_engraver is part of the following context(s) in \layout: ChordGrid (page 69), CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.108 Phrasing\_slur\_engraver

Print phrasing slurs. Similar to Slur\_engraver (page 497).

Music types accepted: note-event (page 55), and phrasing-slur-event (page 56),

This engraver creates the following layout object(s): PhrasingSlur (page 670).

Phrasing\_slur\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.109 Piano\_pedal\_align\_engraver

Align piano pedal symbols and brackets.

Properties (read)

currentCommandColumn (graphical (layout) object)

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

This engraver creates the following layout object(s): SostenutoPedalLineSpanner (page 692), SustainPedalLineSpanner (page 711), and UnaCordaPedalLineSpanner (page 740).

Piano\_pedal\_align\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.110 Piano\_pedal\_engraver

Engrave piano pedal symbols and brackets.

Music types accepted: sostenuto-event (page 57), sustain-event (page 59), and una-corda-event (page 60),

## 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 672), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `UnaCordaPedal` (page 739).

`Piano_pedal_engraver` is part of the following context(s) in `\layout`: `Dynamics` (page 130), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.111 `Piano_pedal_performer`

Music types accepted: `sostenuto-event` (page 57), `sustain-event` (page 59), and `una-corda-event` (page 60),

`Piano_pedal_performer` is part of the following context(s) in `\midi`: `ChordNames` (page 98), `CueVoice` (page 100), `DrumVoice` (page 120), `Dynamics` (page 130), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.112 `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 247), `RhythmicStaff` (page 277), and `StandaloneRhythmStaff` (page 349).

### 2.2.113 Pitched\_trill\_engraver

Print the bracketed note head after a note head with trill.

This engraver creates the following layout object(s): TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), and TrillPitchParentheses (page 733).

Pitched\_trill\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.114 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: ChordNames (page 98), DrumStaff (page 111), GregorianTranscriptionLyrics (page 141), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), Lyrics (page 218), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), VaticanaLyrics (page 388), and VaticanaStaff (page 414).

### 2.2.115 Repeat\_acknowledge\_engraver

This translator adds entries to repeatCommands for events generated by \\repeat volta.

Music types accepted: volta-repeat-end-event (page 60), and volta-repeat-start-event (page 60),

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 *return-count*

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

start-repeat *repeat-count*

Start a repeated section. *repeat-count* is the number of times to perform this 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 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.116 Repeat\_tie\_engraver

Create repeat ties.

Music types accepted: repeat-tie-event (page 56),

This engraver creates the following layout object(s): RepeatTie (page 676), and RepeatTieColumn (page 677).

Repeat\_tie\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207),

MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.117 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 679).

Rest\_collision\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.118 Rest\_engraver

Engrave rests.

Music types accepted: rest-event (page 56),

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 678).

Rest\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.119 Rhythmic\_column\_engraver

Generate NoteColumn, an object that groups stems, note heads, and rests.

This engraver creates the following layout object(s): NoteColumn (page 657).

Rhythmic\_column\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.120 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 680).

Script\_column\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).



### 2.2.121 Script\_engraver

Handle note scripted articulations.

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

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 679).

Script\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), Dynamics (page 130), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.122 Script\_row\_engraver

Determine order in horizontal side position elements.

This engraver creates the following layout object(s): ScriptRow (page 681).

Script\_row\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.123 Separating\_line\_group\_engraver

Generate objects for computing spacing parameters.

Properties (read)

createSpacing (boolean)

Create StaffSpacing objects? Should be set for staves.

Properties (write)

hasStaffSpacing (boolean)

True if currentCommandColumn contains items that will affect spacing.

This engraver creates the following layout object(s): StaffSpacing (page 700).

Separating\_line\_group\_engraver is part of the following context(s) in \layout: ChordNames (page 98), DrumStaff (page 111), FiguredBass (page 135), FretBoards (page 136), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), NoteNames (page 245), PetrucciStaff (page 250), RhythmicStaff (page 277), Staff (page 309), StandaloneRhythmStaff (page 349), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.124 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 575), and ControlPolygon (page 576).

Show\_control\_points\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.125 Signum\_repetitionis\_engraver

Create a SignumRepetitionis at the end of a \repeat volta section.

Music types accepted: volta-repeat-end-event (page 60),

This engraver creates the following layout object(s): SignumRepetitionis (page 685).

Signum\_repetitionis\_engraver is part of the following context(s) in \layout: InternalMensuralStaff (page 180), MensuralStaff (page 221), and PetrucciStaff (page 250).

### 2.2.126 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 696).

Skip\_typesetting\_engraver is part of the following context(s) in \layout: DrumStaff (page 111), GregorianTranscriptionStaff (page 144), InternalGregorianStaff (page 167), InternalMensuralStaff (page 180), KievanStaff (page 194), MensuralStaff (page 221), PetrucciStaff (page 250), Staff (page 309), TabStaff (page 365), and VaticanaStaff (page 414).

### 2.2.127 Slash\_repeat\_engraver

Make beat repeats.

Music types accepted: repeat-slash-event (page 56),

This engraver creates the following layout object(s): DoubleRepeatSlash (page 592), and RepeatSlash (page 675).

Slash\_repeat\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.128 Slur\_engraver

Build slur grobs from slur events.

Music types accepted: note-event (page 55), and slur-event (page 57),

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 688).

Slur\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), NullVoice (page 247), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), and Voice (page 439).

### 2.2.129 Slur\_performer

Music types accepted: slur-event (page 57),

Slur\_performer is part of the following context(s) in \midi: ChordNames (page 98), CueVoice (page 100), DrumVoice (page 120), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), NullVoice (page 247), PetrucciVoice (page 265), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

### 2.2.130 Spacing\_engraver

Make a SpacingSpanner and do bookkeeping of shortest starting and playing notes.

Music types accepted: spacing-section-event (page 57),

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 (non-negative exact rational or +inf.0)

Global override for shortest-playing duration. This is used for switching on proportional notation.

This engraver creates the following layout object(s): SpacingSpanner (page 693).

Spacing\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.131 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 535).

Span\_arpeggio\_engraver is part of the following context(s) in \layout: ChoirStaff (page 67), GrandStaff (page 139), PianoStaff (page 275), and StaffGroup (page 322).

### 2.2.132 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 694).

Span\_bar\_engraver is part of the following context(s) in \layout: GrandStaff (page 139), PianoStaff (page 275), and StaffGroup (page 322).

### 2.2.133 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 695).

Span\_bar\_stub\_engraver is part of the following context(s) in \layout: ChoirStaff (page 67), GrandStaff (page 139), PianoStaff (page 275), and StaffGroup (page 322).

### 2.2.134 `Span_stem_engraver`

Connect cross-staff stems to the stems above in the system

This engraver creates the following layout object(s): `Stem` (page 703).

`Span_stem_engraver` is not part of any context

### 2.2.135 `Spanner_break_forbid_engraver`

Forbid breaks in certain spanners.

`Spanner_break_forbid_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.136 `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 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 2.2.137 `Staff_collecting_engraver`

Maintain the `stavesFound` variable.

Properties (read)

`stavesFound` (list of grobs)

A list of all staff-symbols found.

Properties (write)

`stavesFound` (list of grobs)

A list of all staff-symbols found.

`Staff_collecting_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `Score` (page 283), `Staff` (page 309), `StandaloneRhythmScore` (page 323), `TabStaff` (page 365), `VaticanaScore` (page 390), and `VaticanaStaff` (page 414).

### 2.2.138 `Staff_highlight_engraver`

Highlights music passages.

Music types accepted: `staff-highlight-event` (page 58),

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 699).

`Staff_highlight_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `StandaloneRhythmStaff` (page 349), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.139 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`: `ChordGrid` (page 69), `ChordNames` (page 98), `DrumStaff` (page 111), `GregorianTranscriptionLyrics` (page 141), `GregorianTranscriptionStaff` (page 144), `KievanStaff` (page 194), `Lyrics` (page 218), `MensuralStaff` (page 221), `NoteNames` (page 245), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.140 Staff\_symbol\_engraver

Create the constellation of five (default) staff lines.

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

This engraver creates the following layout object(s): `StaffSymbol` (page 701).

`Staff_symbol_engraver` is part of the following context(s) in `\layout`: `ChordGrid` (page 69), `DrumStaff` (page 111), `GregorianTranscriptionStaff` (page 144), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `KievanStaff` (page 194), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), `StandaloneRhythmStaff` (page 349), `TabStaff` (page 365), and `VaticanaStaff` (page 414).

### 2.2.141 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 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

### 2.2.142 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 702).

`Stanza_number_engraver` is part of the following context(s) in `\layout`: `GregorianTranscriptionLyrics` (page 141), `Lyrics` (page 218), and `VaticanaLyrics` (page 388).

### 2.2.143 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 59),

**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 605), `Stem` (page 703), `StemStub` (page 705), and `StemTremolo` (page 705).

`Stem_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), and `Voice` (page 439).

**2.2.144 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 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

`System_start_delimiter_engraver` is part of the following context(s) in `\layout`: `ChoirStaff` (page 67), `ChordGrid` (page 69), `GrandStaff` (page 139), `PianoStaff` (page 275), `Score` (page 283), `StaffGroup` (page 322), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

**2.2.145 Tab\_note\_heads\_engraver**

Generate one or more tablature note heads from event of type `NoteEvent`.

Music types accepted: `fingering-event` (page 52), `note-event` (page 55), and `string-number-event` (page 58),

**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 717).

`Tab_note_heads_engraver` is part of the following context(s) in `\layout: TabVoice` (page 377).

### 2.2.146 `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 701).

`Tab_staff_symbol_engraver` is part of the following context(s) in `\layout: TabStaff` (page 365).

### 2.2.147 `Tab_tie_follow_engraver`

Adjust `TabNoteHead` properties when the `TabNoteHead` holds a `RepeatTie`, when a `Tie` ends and when a `Slur` or `Glissando` starts at a tied `TabNoteHead`.

Properties (read)

`tabFullNotation` (boolean)

Flag whether `\tabFullNotation` is used

`Tab_tie_follow_engraver` is part of the following context(s) in `\layout: TabVoice` (page 377).

**2.2.148 Tempo\_performer**

Properties (read)

tempoWholesPerMinute (positive exact rational or +inf.0)

The tempo in whole notes per minute.

Tempo\_performer is part of the following context(s) in \midi: ChordGridScore (page 74), and Score (page 283).

**2.2.149 Text\_engraver**

Create text scripts.

Music types accepted: text-script-event (page 59),

This engraver creates the following layout object(s): TextScript (page 721).

Text\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), Dynamics (page 130), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), VaticanaVoice (page 429), and Voice (page 439).

**2.2.150 Text\_mark\_engraver**

Engraves arbitrary textual marks.

Music types accepted: text-mark-event (page 59),

Properties (read)

stavesFound (list of grobs)

A list of all staff-symbols found.

This engraver creates the following layout object(s): TextMark (page 719).

Text\_mark\_engraver is part of the following context(s) in \layout: ChordGridScore (page 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

**2.2.151 Text\_spanner\_engraver**

Create text spanner from an event.

Music types accepted: text-span-event (page 59),

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 723).

Text\_spanner\_engraver is part of the following context(s) in \layout: CueVoice (page 100), DrumVoice (page 120), Dynamics (page 130), GregorianTranscriptionVoice (page 157), KievanVoice (page 207), MensuralVoice (page 235), PetrucciVoice (page 265), StandaloneRhythmVoice (page 354), TabVoice (page 377), and Voice (page 439).

**2.2.152 Tie\_engraver**

Generate ties between note heads of equal pitch.

Music types accepted: tie-event (page 59),

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 725), and `TieColumn` (page 727).

`Tie_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `NoteNames` (page 245), `NullVoice` (page 247), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.153 Tie\_performer

Generate ties between note heads of equal pitch.

Music types accepted: `tie-event` (page 59),

Properties (read)

`tieWaitForNote` (boolean)

If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

Properties (write)

`tieMelismaBusy` (boolean)

Signal whether a tie is present.

`Tie_performer` is part of the following context(s) in `\midi`: `ChordNames` (page 98), `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `NullVoice` (page 247), `PetrucchiVoice` (page 265), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.154 Time\_signature\_engraver

Create a `TimeSignature` (page 727), whenever `timeSignatureFraction` changes.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`initialTimeSignatureVisibility` (vector)

break visibility for the initial time signature.

`partialBusy` (boolean)

Signal that `\partial` acts at the current timestep.

`timeSignatureFraction` (simple time signature)

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 727).

`Time_signature_engraver` is part of the following context(s) in `\layout`: `DrumStaff` (page 111), `InternalGregorianStaff` (page 167), `InternalMensuralStaff` (page 180), `MensuralStaff` (page 221), `PetrucchiStaff` (page 250), `RhythmicStaff` (page 277), `Staff` (page 309), and `TabStaff` (page 365).

### 2.2.155 Time\_signature\_performer

Creates a MIDI time signature whenever `timeSignatureFraction` changes or a `\time` command is issued.

Music types accepted: `time-signature-event` (page 59),

Properties (read)

`timeSignatureFraction` (simple time signature)

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`: `ChordGridScore` (page 74), and `Score` (page 283).

### 2.2.156 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 49), `bar-check-event` (page 50), `bar-event` (page 50), `fine-event` (page 52), and `partial-event` (page 55),

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.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`timeSignatureFraction` (simple time signature)

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 first volta number for the current `\alternative` element. Not set outside of alternatives.

`beatBase` (positive exact rational or `+inf.0`)

The musical length corresponding to one unit of `beatStructure`.

`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` (positive exact rational or `+inf.0`)

The musical length of the current measure.

`measurePosition` (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`timeSignatureFraction` (simple time signature)

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 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390); in `\midi`: `ChordGridScore` (page 74), and `Score` (page 283).

### 2.2.157 `Trill_spanner_engraver`

Create trill spanners.

Music types accepted: `trill-span-event` (page 60),

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 734).

`Trill_spanner_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.158 `Tuplet_engraver`

Catch tuplet events and generate appropriate bracket.

Music types accepted: `tuplet-span-event` (page 60),

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 736), and `TupletNumber` (page 737).

`Tuplet_engraver` is part of the following context(s) in `\layout`: `CueVoice` (page 100), `DrumVoice` (page 120), `GregorianTranscriptionVoice` (page 157), `KievanVoice` (page 207), `MensuralVoice` (page 235), `PetrucchiVoice` (page 265), `StandaloneRhythmVoice` (page 354), `TabVoice` (page 377), `VaticanaVoice` (page 429), and `Voice` (page 439).

### 2.2.159 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 74), Score (page 283), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.160 Vaticana\_ligature\_engraver

Handle ligatures by glueing special ligature heads together.

Music types accepted: ligature-event (page 53), and pes-or-flexa-event (page 56),

This engraver creates the following layout object(s): DotColumn (page 588), and VaticanaLigature (page 741).

Vaticana\_ligature\_engraver is part of the following context(s) in \layout: VaticanaVoice (page 429).

### 2.2.161 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 698), and VerticalAlignment (page 742).

Vertical\_align\_engraver is part of the following context(s) in \layout: ChoirStaff (page 67), ChordGridScore (page 74), GrandStaff (page 139), PianoStaff (page 275), Score (page 283), StaffGroup (page 322), StandaloneRhythmScore (page 323), and VaticanaScore (page 390).

### 2.2.162 Volta\_engraver

Make volta brackets.

Music types accepted: dal-segno-event (page 51), fine-event (page 52), and volta-span-event (page 60),

Properties (read)

currentCommandColumn (graphical (layout) object)

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

printTrivialVoltaRepeats (boolean)

Notate volta-style repeats even when the repeat count is 1.

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 return-count

End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat repeat-count`

Start a repeated section. *repeat-count* is the number of times to perform this 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.

This engraver creates the following layout object(s): `VoltaBracket` (page 745), and `VoltaBracketSpanner` (page 747).

`Volta_engraver` is part of the following context(s) in `\layout`: `ChordGridScore` (page 74), `Score` (page 283), `StandaloneRhythmScore` (page 323), and `VaticanaScore` (page 390).

## 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)`

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.

`alternativeNumber (non-negative, exact integer)`

When set, the first volta number for the current `\alternative` element. Not set outside of alternatives.

`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 *#t* then beams are generated automatically. If set to *#f*, auto-beaming is switched off as soon as the current beam (if any) is finished according to the auto-beaming rules.

`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.

`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.

`beamMaximumSubdivision (non-negative exact rational or +inf.0)`

Maximum interval to subdivide beams, limiting the depth of beamlets removed from subdivision. Ranges from 0 to infinity ( 0=no subdivision, +inf.0=no limit).

`beamMinimumSubdivision (non-negative exact rational or +inf.0)`

Minimum interval to subdivide beams, ignoring beamlets whose subdivision depth is too shallow. Ranges from 0 to infinity ( 0=no limit, +inf.0=no subdivision).

`beatBase (positive exact rational or +inf.0)`

The musical length corresponding to one unit of `beatStructure`.

`beatExtraVelocity (integer)`

Extra MIDI velocity added by the 'Beat\_performer' at the start of each beat.

`beatStructure (list)`

A sequence describing the length of each beat in the measure in units of `beatBase`.

`breathMarkType (symbol)`

The type of `BreathingSign` to create at `\breathe`.

`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.

`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.

`chordNoteNamer` (procedure)

A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)

The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)

A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)

Name of the symbol within the music font.

`clefPosition` (number)

Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)

Add this much extra transposition to a clef. Values of 7 and -7 are common.

`clefTranspositionFormatter` (procedure)

A procedure that takes the transposition number of a `Clef` grob as a string and the style as a symbol and returns a markup.

`clefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a clef is displayed. Possible values are 'default, 'parenthesized, and 'bracketed.



`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.

`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` (positive exact rational or `+inf.0`)

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 to a cue clef. Values of 7 and -7 are common.

`cueClefTranspositionFormatter` (procedure)

A procedure that takes the transposition number of a `cueClef` grob as a string and the style as a symbol and returns a markup.

`cueClefTranspositionStyle` (symbol)

Determines the way the `ClefModifier` grob of a cue clef 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 *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*.

`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.’.

`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 `\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. | :’.

`doubleSlurs` (boolean)

If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)

A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)

A hash table which maps drums to layout settings. Predefined values: ‘drums-style’, ‘agostini-drums-style’, ‘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.

`endAtSkip` (boolean)

End `DurationLine` grob on skip-event

`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’.

`explicitClefVisibility` (vector)

‘break-visibility’ function for clef changes.

`explicitCueClefVisibility` (vector)

‘break-visibility’ function for cue clef changes.

`explicitKeySignatureVisibility` (vector)

‘break-visibility’ function for explicit key changes. ‘\override’ of the break-visibility property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)

Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)

Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`figuredBassAlterationDirection` (direction)

Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)

Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)

A routine generating a markup for a bass figure.

`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`.

`fingeringOrientations` (list)

A list of symbols, containing ‘left’, ‘right’, ‘up’ and/or ‘down’. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)

If true, create a new clef when starting a staff.

`followVoice` (boolean)

If set, note heads are tracked across staff switches by a thin line.

fontSize (number)

The relative size of all grobs in a context.

forbidBreak (boolean)

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

forbidBreakBetweenBarLines (boolean)

If set to #t, 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 (positive exact rational or +inf.0)

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 bar number. 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` (positive exact rational or +inf.0)

The musical length of the current 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`.

`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 where this is set. In addition to suppressing 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.

`pageTurnMinimumRepeatLength` (non-negative exact rational or `+inf.0`)

Minimum length of a repeated section for a page turn to be allowed within that section.

`pageTurnMinimumRestLength` (non-negative exact rational or `+inf.0`)

Minimum length of a rest for a page turn to be allowed.

`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.

`printInitialRepeatBar` (boolean)  
Use a special bar line at the start of a volta repeat even at the beginning of the piece.

`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?

`printTrivialVoltaRepeats` (boolean)  
Notate volta-style repeats even when the repeat count is 1.

`proportionalNotationDuration` (non-negative exact rational or `+inf.0`)  
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.

`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` *return-count*  
End a repeated section. *return-count* is the number of times to go back from this point to the beginning of the section.

`start-repeat` *repeat-count*  
Start a repeated section. *repeat-count* is the number of times to perform this 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.

`respectIncompleteBeams` (boolean)  
When subdividing beams, limit the beam subdivision interval until it is less than or equal to the remaining length from the current moment. Note that it is somewhat unclear whether



to treat incomplete beams specially or not in beam subdivision is the correct way of valid notation. The default value of false is said to be the correct option, although beam subdivision as if this property is true is not unpopular.

`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 `#t`, 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.

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.

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, beams of multiple stems may be subdivided by omitting a number of beamlets, dependent on beamMaximumSubdivision, between beats at multiples of beamMinimumSubdivision.

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 `#t`.

`suspendRestMerging` (boolean)

When using the `Merge_rest_engraver` do not merge rests when this is set to `#t`.

`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.

`tabFullNotation` (boolean)

Flag whether `\tabFullNotation` is used

`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` (positive exact rational or `+inf.0`)

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` (simple time signature)

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 `beatBase`, `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` (non-negative exact rational or `+inf.0`)

Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. This property can shorten the bracket.

```
{
 \set tupletSpannerDuration = #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.

`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.

`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`.

`currentTupletDescription` (ly:tuplet-description)

An object describing the current tuplet description, or '(). Tuplet description objects are opaque to Scheme.

`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.

`measurePosition` (moment)

The current point within the measure.

`measureStartNow` (boolean)

True at the beginning of a measure.

`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.

`propertyStacks` (association list (list of pairs))

An alist used by `\pushContextProperty` and `\popContextProperty` to emulate a context property stack. Each element of a stack is a list: either an empty list representing the unset state or a one-element list holding the value for the set state.

`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.

`voltaBracketMusicalLength` (non-negative moment with no grace part)

The maximum musical length of a `VoltaBracket` when its `musical-length` property is not set.

This property is deprecated; overriding the `musical-length` property of `VoltaBracket` is recommended.

## 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 528), `grob`.

Accidental objects are created by the following engraver(s): `Accidental_engraver` (page 450).

Standard settings:

`after-line-breaking` (boolean):

`ly:accidental-interface::remove-tied`

Dummy property, used to trigger callback for `after-line-breaking`.

`alteration` (number):

`accidental-interface::calc-alteration`

Alteration numbers for accidental.

`avoid-slur` (symbol):

`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`extra-spacing-width` (pair of numbers):

`'(-0.2 . 0.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the ‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`glyph-name` (string):

`accidental-interface::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`horizontal-skylines` (pair of skylines):

`#<unpure-pure-container 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 ly:grob::vertical-skylines-from-stencil`

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

Y-extent (pair of numbers):

`#<unpure-pure-container 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 749), `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `inline-accidental-interface` (page 787), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.2 AccidentalCautionary

A cautionary accidental, normally enclosed in parentheses.

AccidentalCautionary objects are created by the following engraver(s): `Accidental_engraver` (page 450).

Standard settings:

`after-line-breaking` (boolean):

`ly:accidental-interface::remove-tied`

Dummy property, used to trigger callback for `after-line-breaking`.

`alteration` (number):

`accidental-interface::calc-alteration`

Alteration numbers for accidental.

`avoid-slur` (symbol):

`'inside`

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`extra-spacing-width` (pair of numbers):

`'(-0.2 . 0.0)`

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`glyph-name` (string):

`accidental-interface::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.



`horizontal-skylines` (pair of skylines):  
`#<unpure-pure-container 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):  
`ly:accidental-interface::print`  
 The symbol to print.

`vertical-skylines` (pair of skylines):  
`#<unpure-pure-container ly:grob::vertical-skylines-from-stencil`  
`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.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):  
`#<unpure-pure-container 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 749), `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `inline-accidental-interface` (page 787), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.3 AccidentalPlacement

In groups of `Accidental` (page 526), 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 the following engraver(s): `Accidental_engraver` (page 450), and `Ambitus_engraver` (page 451).

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 750), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.4 AccidentalSuggestion

An annotational accidental as used in *musica ficta*. Normally positioned above a note.

`AccidentalSuggestion` objects are created by the following engraver(s): `Accidental_engraver` (page 450).

Standard settings:

after-line-breaking (boolean):

ly:accidental-interface::remove-tied

Dummy property, used to trigger callback for after-line-breaking.

alteration (number):

accidental-interface::calc-alteration

Alteration numbers for accidental.

direction (direction):

1

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

font-size (number):

-2

The font size, compared to the 'normal' size. 0 is style-sheet's normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

glyph-name (string):

accidental-interface::calc-glyph-name

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `accidental-interface` (page 749), `accidental-suggestion-interface` (page 751), `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `script-interface` (page 814), `self-alignment-interface` (page 815), and `side-position-interface` (page 818).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.5 Ambitus

An ambitus, giving the range of pitches of a voice or instrument. It aligns `AmbitusAccidental` (page 533), `AmbitusLine` (page 533), and `AmbitusNoteHead` (page 534), horizontally and defines the horizontal spacing from the ambitus to other items.

Ambitus objects are created by the following engraver(s): `Ambitus_engraver` (page 451).

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 #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):

```
'((cue-end-clef extra-space . 0.5)
 (clef extra-space . 1.15)
 (cue-clef extra-space . 0.5)
 (key-signature extra-space . 1.15)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

- extra-space*  
Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.
- shrink-space*  
Only compatible with *first-note* and *next-note*. Put this much space between the two grobs. The space is only shrinkable.
- semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

X-extent (pair of numbers):

`ly:axis-group-interface::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

Y-extent (pair of numbers):

`#<unpure-pure-container 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.

This object supports the following interface(s): `ambitus-interface` (page 752), `axis-group-interface` (page 753), `break-aligned-interface` (page 762), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.6 AmbitusAccidental

An accidental in an `Ambitus` (page 531).

`AmbitusAccidental` objects are created by the following engraver(s): `Ambitus_engraver` (page 451).

Standard settings:

`glyph-name` (string):

`accidental-interface::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

Y-extent (pair of numbers):

`#<unpure-pure-container 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 749), `accidental-switch-interface` (page 751), `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.7 AmbitusLine

The vertical line in an `Ambitus` (page 531).

`AmbitusLine` objects are created by the following engraver(s): `Ambitus_engraver` (page 451).

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the *self-alignment-interface* (page 815).

This object supports the following interface(s): *ambitus-interface* (page 752), *font-interface* (page 774), *grob-interface* (page 780), and *item-interface* (page 790).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.8 AmbitusNoteHead

A note head in an *Ambitus* (page 531).

*AmbitusNoteHead* objects are created by the following engraver(s): *Ambitus\_engraver* (page 451).

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 or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`ignore-ambitus` (boolean):  
     #t  
     If set, don't consider this notehead for ambitus calculation.

`stencil` (stencil):  
     `ly:note-head::print`  
     The symbol to print.

`Y-extent` (pair of numbers):  
     #<unpure-pure-container 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 ly:staff-symbol-referencer::callback >  
     The vertical amount that this object is moved relative to its Y-parent.  
     Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `ambitus-interface` (page 752), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `ledgered-interface` (page 793), `note-head-interface` (page 805), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.9 Arpeggio

An arpeggio line (normally a vertical wiggle).

Arpeggio objects are created by the following engraver(s): `Arpeggio_engraver` (page 452), and `Span_arpeggio_engraver` (page 498).

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 a bracket indicating non-arpeggio or optional material, the length of the horizontal edges.

`script-priority (number):`

0

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`side-axis (number):`

0

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

`staff-position (number):`

0.0

Vertical position, measured in half staff spaces, counted from the middle line.

For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

`stencil (stencil):`

`ly:arpeggio::print`

The symbol to print.

`thickness (number):`

1

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`X-extent (pair of numbers):`

`ly:arpeggio::width`

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset (number):`

`ly:side-position-interface::x-aligned-side`

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`

`#<unpure-pure-container ly:grob::stencil-height 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `arpeggio-interface` (page 752), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `side-position-interface` (page 818), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.10 BalloonText

A balloon text with a pointing line to visually mark and annotate another grob.

BalloonText objects are created by the following engraver(s): `Balloon_engraver` (page 453).

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):

`#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 at lily/output-lib.scm:3567:0 (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 at lily/output-lib.scm:1661:0 (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 ly:grob::vertical-skylines-from-stencil
 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 at lily/output-lib.scm:1661:0 (grob)>
```

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):

```
#<unpure-pure-container ly:grob::stencil-height 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 at lily/output-lib.scm:1661:0 (grob)>
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `balloon-interface` (page 755), `font-interface` (page 774), `grob-interface` (page 780), `sticky-grob-interface` (page 833), and `text-interface` (page 837).

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 790), and `spanner-interface` (page 826).

### 3.1.11 BarLine

A bar line.

BarLine objects are created by the following engraver(s): `Bar_engraver` (page 453).

Standard settings:

`allow-span-bar` (boolean):

```
#t
```

If false, no inter-staff bar line will be created below this bar line.

`bar-extent` (pair of numbers):

```
ly:bar-line::calc-bar-extent
```

The Y-extent of the actual bar line. This may differ from Y-extent because it does not include the dots in a repeat bar line.

`break-align-anchor` (number):

`ly:bar-line::calc-anchor`

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-symbol` (symbol):

`'staff-bar`

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in *Internals Reference*.

`break-visibility` (vector):

`bar-line::calc-break-visibility`

A vector of 3 booleans,  `#(end-of-line unbroken begin-of-line)`. `#t` means visible, `#f` means killed.

`extra-spacing-height` (pair of numbers):

`pure-from-neighbor-interface::account-for-span-bar`

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`gap` (dimension, in staff space):

0.4

Size of a gap in a variable symbol.

`glyph` (string):

`"|"`

A string determining what ‘style’ of glyph is typeset. Valid choices depend on the function that is reading this property.

In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-left` (string):

`#<procedure at lily/output-lib.scm:1761:0 (grob)>`

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 or clefs, *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 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*.

right-justified (boolean):

```
#f
```

Used for *BarLines* to right-align them. Usually the extent of a *BarLine* has some positive value to the right. If this property is set to #t, *BarLine.stencil* is translated to the left by this value. Needs to be set at *Score* or *StaffGroup* level. As a result all *BarLines* of said *Score* or *StaffGroup* are right-justified.

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*).

short-bar-extent (pair of numbers):

```
ly:bar-line::calc-short-bar-extent
```

The Y-extent of a short bar line. The default is half the normal bar extent, rounded up to an integer number of staff spaces.

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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0))
```

```
(first-note semi-shrink-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)
```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```
extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable when paired with first-note or next-note; other-
 wise it is fixed.

minimum-space
 Put at least this much space between the left sides of both grobs, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

shrink-space
 Only compatible with first-note and next-note. Put this much
 space between the two grobs. The space is only shrinkable.
```

`semi-shrink-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 shrinkable.

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 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 755), `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), and `pure-from-neighbor-interface` (page 812).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.12 BarNumber

An ordinary bar number. Centered bar numbers are managed separately with `CenteredBarNumber` (page 561), grobs.

`BarNumber` objects are created by the following engraver(s): `Bar_number_engraver` (page 456).

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-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 at lily/output-lib.scm:509:2 (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. If not a number, align on the object's reference point.

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.



Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): bar-number-interface (page 757), break-alignable-interface (page 762), font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.13 BassFigure

A number in figured bass. It can contain an alteration as well.

BassFigure objects are created by the following engraver(s): Figured\_bass\_engraver (page 471).

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 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 751), bass-figure-interface (page 757), font-interface (page 774), grob-interface (page 780), item-interface (page 790), rhythmic-grob-interface (page 813), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.14 BassFigureAlignment

An auxiliary grob to stack several BassFigureLine (page 548), grobs vertically.

BassFigureAlignment objects are created by the following engraver(s): Figured\_bass\_engraver (page 471).

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 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.

This object supports the following interface(s): `align-interface` (page 751), `axis-group-interface` (page 753), `bass-figure-alignment-interface` (page 757), `grob-interface` (page 780), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.15 BassFigureAlignmentPositioning

If figured bass is used in the `Staff` (page 309), 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 the following engraver(s): `Figured_bass_position_engraver` (page 472).

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.

outside-staff-priority (number):

25

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.

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 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):

#<unpure-pure-container ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), grob-interface (page 780), outside-staff-interface (page 808), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.16 BassFigureBracket

Brackets around a figured bass (or elements of it).

BassFigureBracket objects are created by the following engraver(s): Figured\_bass\_engraver (page 471).

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 771), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 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 the following engraver(s): `Figured_bass_engraver` (page 471).

Standard settings:

`bound-details` (alist, with symbols as keys):

```
'((right (attach-dir . 1) (padding . -0.15))
 (right-broken (attach-dir . -1) (padding . 0.5))
 (left-broken (attach-dir . 1) (padding . 0.5))
 (left (attach-dir . 1) (padding . 0.15)))
```

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.

`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.

`stencil` (`stencil`):

`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.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `figured-bass-continuation-interface` (page 772), `grob-interface` (page 780), `horizontal-line-spanner-interface` (page 786), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.18 BassFigureLine

An auxiliary grob providing a baseline for bass figures that should be aligned vertically.

BassFigureLine objects are created by the following engraver(s):  
Figured\_bass\_engraver (page 471).

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::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 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.

This object supports the following interface(s): axis-group-interface (page 753), grob-interface (page 780), outside-staff-axis-group-interface (page 807), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.19 Beam

A beam.

Beam objects are created by the following engraver(s): `Auto_beam_engraver` (page 452), `Beam_engraver` (page 457), `Chord_tremolo_engraver` (page 462), `Grace_auto_beam_engraver` (page 474), and `Grace_beam_engraver` (page 475).

Standard settings:

`accidental-padding` (number):

1.0

Property used by Beam to avoid accidentals in whole note tremolos.

`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-accidental-interface  
key-signature-interface  
note-head-interface  
stem-interface  
time-signature-interface)

A list of interfaces for which automatic beam-collision resolution is run.

`damping` (number):

1

Amount of beam slope damping.

`details` (alist, with symbols as keys):

'((beam-eps . 0.001)  
(collision-padding . 0.35)

```
(collision-penalty . 500)
(damping-direction-penalty . 800)
(hint-direction-penalty . 20)
(ideal-slope-factor . 10)
(musical-direction-factor . 400)
(over-beam-height . 0.75)
(region-size . 2)
(round-to-zero-slope . 0.02)
(secondary-beam-demerit . 10)
(slash-slope . 2)
(slash-side . -1)
(slash-stem-fraction . 0.3)
(slash-thickness . 0.1)
(slash-X-positions -0.5 . 1)
(stem-collision-factor . 0.1)
(stem-length-demerit-factor . 5)
(stem-length-limit-penalty . 5000))
```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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-size (number):

```
-6
```

The font size, compared to the 'normal' size. 0 is style-sheet's normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

gap (dimension, in staff space):

```
0.8
```

Size of a gap in a variable symbol.

knee (boolean):

```
ly:beam::calc-knee
```

Is this beam kneed?

minimum-length (dimension, in staff space):

```
6.0
```

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.

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 (*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.

springs-and-rods (boolean):

ly:beam::tremolo-springs-and-rods

Dummy variable for triggering spacing routines.

stencil (stencil):

ly:beam::print

The symbol to print.

transparent (boolean):

#<procedure at lily/output-lib.scm:1722:0 (grob)>

This makes the grob invisible.

vertical-skylines (pair of skylines):

#<unpure-pure-container ly:grob::vertical-skylines-from-stencil

ly:grob::pure-simple-vertical-skylines-from-extents >

Two skylines, one above and one below this grob.

X-positions (pair of numbers):

ly:beam::calc-x-positions

Pair of X staff coordinates of a spanner in the form (*left* . *right*), where both *left* and *right* are in staff-space units of the current staff.

This object supports the following interface(s): beam-interface (page 757), grob-interface (page 780), spanner-interface (page 826), staff-symbol-referencer-interface (page 829), and unbreakable-spanner-interface (page 845).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.20 BendAfter

A grob for displaying *falls* and *doits*.

BendAfter objects are created by the following engraver(s): `Bend_engraver` (page 459).

Standard settings:

minimum-length (dimension, in staff space):

0.5

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a Tie, this sets the minimum distance between noteheads.

stencil (stencil):

bend::print

The symbol to print.



thickness (number):  
2.0

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

This object supports the following interface(s): *bend-after-interface* (page 760), *grob-interface* (page 780), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.21 BendSpanner

A string bending as used in tablature notation.

*BendSpanner* objects are created by the following engraver(s): *Bend\_spanner\_engraver* (page 459).

Standard settings:

avoid-slur (symbol):  
'ignore

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):  
3

Distance between base lines of multiple lines of text.

before-line-breaking (boolean):  
*bend::target-cautionary*

Dummy property, used to trigger a callback function.

details (alist, with symbols as keys):

```
'((arrow-stencil
.
#<procedure bend::arrow-head-stencil (thickness x-y-coords height width dir)>
(bend-amount-strings
(quarter . " $\frac{1}{4}$ ")
(half . " $\frac{1}{2}$ ")
(three-quarter . " $\frac{3}{4}$ ")
(full . #f))
(bend-arrowhead-height . 1.25)
(bend-arrowhead-width . 0.8)
(curvature-factor . 0.35)
(curve-x-padding-line-end . 0.5)
(curve-y-padding-line-end . 1)
(dashed-line-settings 0.4 0.4 0)
(head-text-break-visibility . #(#f #t #t))
(horizontal-left-padding . 0.1)
(successive-level . 1)
```

```
(target-visibility . #f)
(vertical-padding . 0.2)
(y-distance-from-tabstaff-to-arrow-tip . 2.75))
```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob’s details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob’s description section.

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. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 *Staff.StaffSymbol.thickness*).

vertical-skylines (pair of skylines):

```
#<unpure-pure-container ly:grob::vertical-skylines-from-stencil
ly:grob::pure-simple-vertical-skylines-from-extents >
```

Two skylines, one above and one below this grob.

word-space (dimension, in staff space):

0.6

Space to insert between words in texts.

Y-offset (number):

0

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): bend-interface (page 760), font-interface (page 774), grob-interface (page 780), line-spanner-interface (page 795), outside-staff-interface (page 808), spanner-interface (page 826), text-interface (page 837), and text-script-interface (page 837).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.22 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 *BreakAlignment* (page 555).

*BreakAlignGroup* objects are created by the following engraver(s): *Break\_align\_engraver* (page 460).

Standard settings:

axes (list):

'(0)

List of axis numbers. In the case of alignment grobs, this should contain only one number.

break-align-anchor (number):

ly:break-aligned-interface::calc-average-anchor

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

break-align-anchor-alignment (number):

ly:break-aligned-interface::calc-joint-anchor-alignment

Read by *ly:break-aligned-interface::calc-extent-aligned-anchor* for aligning an anchor to a grob's extent.

break-visibility (vector):

ly:break-aligned-interface::calc-break-visibility

A vector of 3 booleans, *#(end-of-line unbroken begin-of-line)*. #t means visible, #f means killed.

X-extent (pair of numbers):

ly:axis-group-interface::width

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

This object supports the following interface(s): *axis-group-interface* (page 753), *break-aligned-interface* (page 762), *grob-interface* (page 780), and *item-interface* (page 790).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.23 BreakAlignment

An auxiliary grob that manages the horizontal ordering of *BreakAlignGroup* (page 554), grobs within a *NonMusicalPaperColumn* (page 655), grob (for example, whether the time signature follows or precedes a bar line).

*BreakAlignment* objects are created by the following engraver(s): *Break\_align\_engraver* (page 460).

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):

```
#((staff-ellipsis
 left-edge
 cue-end-clef
 ambitus
 breathing-sign
 optional-material-end-bracket
 signum-repetitionis
 clef
 cue-clef
 staff-bar
 key-cancellation
 key-signature
 time-signature
 optional-material-start-bracket
 custos)
(staff-ellipsis
 left-edge
 optional-material-end-bracket
 cue-end-clef
 ambitus
 breathing-sign
 signum-repetitionis
 clef
 cue-clef
```

```

 staff-bar
 key-cancellation
 key-signature
 time-signature
 optional-material-start-bracket
 custos)
(staff-ellipsis
 left-edge
 optional-material-end-bracket
 ambitus
 breathing-sign
 signum-repetitionis
 clef
 key-cancellation
 key-signature
 time-signature
 staff-bar
 cue-clef
 optional-material-start-bracket
 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 753), `break-alignment-interface` (page 764), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.24 BreathingSign

A breathing sign.

BreathingSign objects are created by the following engraver(s):  
`Breathing_sign_engraver` (page 460), and `Caesura_engraver` (page 460).

Standard settings:

`break-align-symbol` (symbol):

`'breathing-sign`

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in *Internals Reference*.

`break-visibility` (vector):

`##(#t #t #f)`

A vector of 3 booleans, `##(end-of-line unbroken begin-of-line)`. `#t` means visible, `#f` means killed.

`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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

- extra-space*  
Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.
- shrink-space*  
Only compatible with *first-note* and *next-note*. Put this much space between the two grobs. The space is only shrinkable.
- semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

*stencil* (*stencil*):

ly:text-interface::print

The symbol to print.

`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 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 ly:breathing-sign::offset-callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `break-aligned-interface` (page 762), `breathing-sign-interface` (page 765), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 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 the following engraver(s): `Caesura_engraver` (page 460).

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. If not a number, align on the object's reference point.

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 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.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):  
 #<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
 ly:side-position-interface::pure-y-aligned-side >  
 The vertical amount that this object is moved relative to its Y-parent.  
 Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `caesura-script-interface` (page 765), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `script-interface` (page 814), `self-alignment-interface` (page 815), and `side-position-interface` (page 818).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.26 CenteredBarNumber

A centered bar number; see also `CenteredBarNumberLineSpanner` (page 562). Ordinary bar numbers are managed with `BarNumber` (page 542), grobs.

`CenteredBarNumber` objects are created by the following engraver(s): `Bar_number_engraver` (page 456).

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-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. If not a number, align on the object's reference point.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `bar-number-interface` (page 757), `centered-bar-number-interface` (page 765), `centered-spanner-interface` (page 766), `font-interface` (page 774), `grob-interface` (page 780), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.27 CenteredBarNumberLineSpanner

An auxiliary grob providing a vertical baseline to align CenteredBarNumber (page 561), grobs.

CenteredBarNumberLineSpanner objects are created by the following engraver(s):  
Centered\_bar\_number\_align\_engraver (page 461).

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 ly:grob::vertical-skylines-from-element-stencils  
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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), bar-number-interface (page 757), centered-bar-number-line-spanner-interface (page 765), grob-interface (page 780), outside-staff-interface (page 808), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.28 ChordName

A stand-alone chord name. For chord names in chord grids, see GridChordName (page 612).

ChordName objects are created by the following engraver(s): Chord\_name\_engraver (page 461).

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 serif, sans and typewriter.

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 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 751), `chord-name-interface` (page 766), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `rhythmic-grob-interface` (page 813), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.29 ChordSquare

In a chord grid, this grob represents one chord square. It helps place `GridChordName` (page 612), 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 701), and `BarLine` (page 538).

`ChordSquare` objects are created by the following engraver(s): `Chord_square_engraver` (page 462).

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)
 (0 . -0.65)
 (0.63 . 0))
 ((1/4 3/4) (-0.63 . 0) (0.38 . 0))
 ((3/4 1/4) (-0.38 . 0) (0.63 . 0)))
```

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 chord-square::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-square-interface` (page 766), `grob-interface` (page 780), `line-interface` (page 794), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.30 Clef

A clef. See also `ClefModifier` (page 568), `CueClef` (page 577), and `CueEndClef` (page 580).

Clef objects are created by the following engraver(s): `Clef_engraver` (page 462).

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 or clefs, *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 extra-space . 0.82)
 (key-signature extra-space . 0.82)
 (time-signature extra-space . 1.52)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)

If `space-alist` is defined for a grob that gets spaced in a staff, an entry for `first-note` must be present. If there is no `next-note` entry, the value of `first-note` is used instead.

Choices for *spacing-style* are:

`extra-space`

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

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 ly:grob::vertical-skylines-from-stencil >`

Two skylines, one above and one below this grob.

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-aligned-interface (page 762), clef-interface (page 767), font-interface (page 774), grob-interface (page 780), item-interface (page 790), pure-from-neighbor-interface (page 812), and staff-symbol-referencer-interface (page 829).

This object is of class Item (characterized by item-interface (page 790)).

### 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 565), CueClef (page 577), and CueEndClef (page 580).

ClefModifier objects are created by the following engraver(s): Clef\_engraver (page 462), and Cue\_clef\_engraver (page 465).

Standard settings:

break-visibility (vector):

```
#<procedure at lily/output-lib.scm:1722:0 (grob)>
```

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):

```
'((G -0.2 . 0.1) (F -0.3 . -0.2) (C 0 . 0))
```

An alist of parent-alignments that should be used for clef modifiers with various clefs

color (color):

```
#<procedure at lily/output-lib.scm:1722:0 (grob)>
```

The color of this grob.

font-shape (symbol):

```
'italic
```

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

font-size (number):

```
-4
```

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

parent-alignment-X (number):

```
ly:clef-modifier::calc-parent-alignment
```

Specify on which point of the parent the object is aligned. The value -1 means aligned on parent’s left edge, 0 on center, and 1 right edge, in X direction. Other numerical values may also be specified - the unit is half the parent’s width. If not a number, align on the parent’s reference point. 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. If not a number, align on the object's reference point.

`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):

`clef-modifier::print`

The symbol to print.

`transparent` (boolean):

`#<procedure at lily/output-lib.scm:1722:0 (grob)>`

This makes the grob invisible.

`vertical-skylines` (pair of skylines):

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side`

`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `clef-modifier-interface` (page 767), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.32 ClusterSpanner

A cluster spanner. The envelope shape within the spanner is given by `ClusterSpannerBeacon` (page 570), grobs.

`ClusterSpanner` objects are created by the following engraver(s): `Cluster_spanner_engraver` (page 463).

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 768), `grob-interface` (page 780), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.33 ClusterSpannerBeacon

An auxiliary grob to specify the minimum and maximum pitch of a `ClusterSpanner` (page 569), grob at a given moment.

`ClusterSpannerBeacon` objects are created by the following engraver(s): `Cluster_spanner_engraver` (page 463).

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 767), `grob-interface` (page 780), `item-interface` (page 790), and `rhythmic-grob-interface` (page 813).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.34 CodaMark

A coda mark.

`CodaMark` objects are created by the following engraver(s): `Mark_engraver` (page 482).

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. If not a number, align on the object's reference point.

stencil (stencil):

ly:text-interface::print

The symbol to print.

vertical-skylines (pair of skylines):

#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-alignable-interface (page 762), coda-mark-interface (page 768), font-interface (page 774), grob-interface (page 780), item-interface (page 790), mark-interface (page 798), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.35 CombineTextScript

A grob for printing markup given in the soloText, soloIIIText, and aDueText properties if automatic part combining is active.

CombineTextScript objects are created by the following engraver(s): Part\_combine\_engraver (page 491).

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. Common choices are `normal` and `bold`. The full list of symbols that can be used is: `thin`, `ultralight` (or `extralight`), `light`, `semilight` (or `demilight`), `book`, `normal` (or `regular`), `medium`, `semibold` (or `demibold`), `bold`, `ultrabold` (or `extrabold`), `heavy` (or `black`), and `ultraheavy` (or `ultrablack` or `extrablack`).

`outside-staff-priority (number):`

`475`

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 not a number, align on the parent’s reference point. 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. If not a number, align on the object's reference point.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`  
#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `text-interface` (page 837), and `text-script-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.36 ControlPoint

A visual representation of a Bézier control point in ties and slurs.

ControlPoint objects are created by the following engraver(s):  
Show\_control\_points\_engraver (page 496).

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  
#t)

Text markup. See Section “Formatting text” in *Notation Reference*.

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.

X-offset (number):  
#<procedure at lily/output-lib.scm:3552:0 (grob)>

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

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 at lily/output-lib.scm:3552:0 (grob)>
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): control-point-interface (page 768), grob-interface (page 780), sticky-grob-interface (page 833), and text-interface (page 837).

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 790), and spanner-interface (page 826).

### 3.1.37 ControlPolygon

A visual representation of a Bézier control polygon as used in ties and slurs.

ControlPolygon objects are created by the following engraver(s):  
Show\_control\_points\_engraver (page 496).

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 768), *grob-interface* (page 780), *sticky-grob-interface* (page 833), and *text-interface* (page 837).

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 790), and *spanner-interface* (page 826).

### 3.1.38 CueClef

A clef starting a cue. See also *Clef* (page 565), *ClefModifier* (page 568), and *CueEndClef* (page 580).

*CueClef* objects are created by the following engraver(s): *Cue\_clef\_engraver* (page 465).

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 or clefs, *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):

```
'((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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)
```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```
extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable when paired with first-note or next-note; other-
 wise it is fixed.

minimum-space
 Put at least this much space between the left sides of both grobs, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

shrink-space
 Only compatible with first-note and next-note. Put this much
 space between the two grobs. The space is only shrinkable.

semi-shrink-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 shrinkable.
```

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 ly:grob::vertical-skylines-from-stencil >

Two skylines, one above and one below this grob.

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-aligned-interface (page 762), clef-interface (page 767), font-interface (page 774), grob-interface (page 780), item-interface (page 790), pure-from-neighbor-interface (page 812), and staff-symbol-referencer-interface (page 829).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.39 CueEndClef

A clef ending a cue. See also Clef (page 565), ClefModifier (page 568), and CueClef (page 577).

CueEndClef objects are created by the following engraver(s): Cue\_clef\_engraver (page 465).

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 ##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 or clefs, *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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)
```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```
extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable when paired with first-note or next-note; other-
 wise it is fixed.

minimum-space
 Put at least this much space between the left sides of both grobs, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

shrink-space
 Only compatible with first-note and next-note. Put this much
 space between the two grobs. The space is only shrinkable.

semi-shrink-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 shrinkable.
```

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

stencil (stencil):

ly:clef::print

The symbol to print.

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-aligned-interface (page 762), clef-interface (page 767), font-interface (page 774), grob-interface (page 780), item-interface (page 790), pure-from-neighbor-interface (page 812), and staff-symbol-referencer-interface (page 829).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.40 Custos

A custos, mainly used in older notation like Gregorian chant.

Custos objects are created by the following engraver(s): Custos\_engraver (page 466).

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)

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

- extra-space*  
Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.
- shrink-space*  
Only compatible with *first-note* and *next-note*. Put this much space between the two grobs. The space is only shrinkable.
- semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

*stencil* (*stencil*):

*ly:custos::print*

The symbol to print.

style (symbol):  
   'vaticana

This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

Y-offset (number):

#<unpure-pure-container ly:staff-symbol-referencer::callback >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-aligned-interface (page 762), custos-interface (page 769), font-interface (page 774), grob-interface (page 780), item-interface (page 790), and staff-symbol-referencer-interface (page 829).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.41 Divisio

A structural divider in a chant, often calling for a breath or caesura.

Divisio objects are created by the following engraver(s): Divisio\_engraver (page 467).

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 aligning 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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 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)
```

If `space-alist` is defined for a grob that gets spaced in a staff, an entry for `first-note` must be present. If there is no `next-note` entry, the value of `first-note` is used instead.

Choices for `spacing-style` are:

**extra-space**

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.

**shrink-space**

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

**semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

**stencil (stencil):**

`ly:text-interface::print`

The symbol to print.

**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 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 ly:breathing-sign::offset-callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-aligned-interface (page 762), breathing-sign-interface (page 765), font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.42 DotColumn

An auxiliary grob to align stacked Dots (page 588), grobs of dotted notes and chords.

DotColumn objects are created by the following engraver(s): Dot\_column\_engraver (page 467), and Vaticana\_ligature\_engraver (page 507).

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.

padding (dimension, in staff space):

dot-column-interface::pad-by-one-dot-width

Add this much extra space between objects that are next to each other.

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 753), dot-column-interface (page 769), grob-interface (page 780), and item-interface (page 790).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.43 Dots

The dot(s) of a dotted note. See also DotColumn (page 588).

Dots objects are created by the following engraver(s): Dots\_engraver (page 468).

Standard settings:

avoid-slur (symbol):

'inside

Method of handling slur collisions. Choices are *inside*, *outside*, *around*, and *ignore*. *inside* adjusts the slur if needed to keep the grob inside the slur. *outside* moves the grob vertically to the outside of the slur. *around* moves the grob vertically to the outside of the slur only if there is a collision. *ignore* does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), *outside* and *around* behave like *ignore*.

`dot-count` (integer):  
`dots::calc-dot-count`  
 The number of dots.

`extra-spacing-height` (pair of numbers):  
`'(-0.5 . 0.5)`  
 In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers):  
`'(0.0 . 0.2)`  
 In the horizontal spacing problem, we pad each item by this amount (by adding the ‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`glyph-name` (string):  
`dots::calc-glyph-name`  
 The glyph name within the font.  
 In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`staff-position` (number):  
`dots::calc-staff-position`  
 Vertical position, measured in half staff spaces, counted from the middle line.  
 For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

`stencil` (stencil):  
`ly:dots::print`  
 The symbol to print.

`Y-extent` (pair of numbers):  
`#<unpure-pure-container 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 770), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.44 DoublePercentRepeat

A double-percent symbol for repeating two bars. See also `DoublePercentRepeatCounter` (page 591), `PercentRepeat` (page 667), `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

`DoublePercentRepeat` objects are created by the following engraver(s):  
`Double_percent_repeat_engraver` (page 468).

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 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 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), and `percent-repeat-interface` (page 810).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.45 DoublePercentRepeatCounter

A grob to print a counter for `DoublePercentRepeat` (page 590), grobs.

`DoublePercentRepeatCounter` objects are created by the following engraver(s): `Double_percent_repeat_engraver` (page 468).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 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 667), DoublePercentRepeat (page 590), and RepeatSlash (page 675).

DoubleRepeatSlash objects are created by the following engraver(s): Slash\_repeat\_engraver (page 497).

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):

#<unpure-pure-container 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 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `percent-repeat-interface` (page 810), and `rhythmic-grob-interface` (page 813).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.47 DurationLine

A horizontal duration line, continuing rhythmic items (usually note heads).

`DurationLine` objects are created by the following engraver(s): `Duration_line_engraver` (page 469).

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):  
'((extra-dot-padding . 0.5)  
  (hook-direction . 1)  
  (hook-height . 0.34)  
  (hook-thickness . #f))

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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 ly:grob::vertical-skylines-from-stencil
 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.
 Note that many objects have special positioning considerations, which cause any
 setting of Y-offset to be ignored or modified, even though the object supports the
 self-alignment-interface (page 815).

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 770), *font-interface* (page 774), *grob-interface* (page 780), *horizontal-line-spanner-interface* (page 786), *line-interface* (page 794), *spanner-interface* (page 826), and *unbreakable-spanner-interface* (page 845).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.48 **DynamicLineSpanner**

An auxiliary grob providing a vertical baseline to align successive dynamic grobs (*DynamicText* (page 597), *DynamicTextSpanner* (page 599), and *Hairpin* (page 614)) within a staff.

*DynamicLineSpanner* objects are created by the following engraver(s): *Dynamic\_align\_engraver* (page 469).

Standard settings:

axes (list):

'(1)

List of axis numbers. In the case of alignment grobs, this should contain only one number.

direction (direction):

-1

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

minimum-space (dimension, in staff space):

1.2

Minimum distance that the victim should move (after padding).

outside-staff-priority (number):

250

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller outside-staff-priority is closer to the staff.

padding (dimension, in staff space):

0.6

Add this much extra space between objects that are next to each other.

side-axis (number):

1

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

slur-padding (number):

0.3

Extra distance between slur and script.

staff-padding (dimension, in staff space):

0.1

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

vertical-skylines (pair of skylines):

```
#<unpure-pure-container ly:grob::vertical-skylines-from-element-stencils
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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), dynamic-interface (page 771), dynamic-line-spanner-interface (page 771), grob-interface (page 780), outside-staff-interface (page 808), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.49 DynamicText

A dynamic text item like ‘ff’ or ‘mp’. See also DynamicLineSpanner (page 595).

DynamicText objects are created by the following engraver(s): Dynamic\_engraver (page 470).

Standard settings:

direction (direction):

```
ly:script-interface::calc-direction
```

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

extra-spacing-width (pair of numbers):

```
'(+inf.0 . -inf.0)
```

In the horizontal spacing problem, we pad each item by this amount (by adding the ‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to (+inf.0 . -inf.0).

font-encoding (symbol):

```
'fetaText
```

The font encoding is the broadest category for selecting a font. Currently, only Lilypond’s system fonts (Emmentaler) are using this property. Available values are fetaMusic (Emmentaler), fetaBraces, fetaText (Emmentaler).

font-series (symbol):

```
'bold
```

Select the series of a font. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

font-shape (symbol):

```
'italic
```

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

stencil (stencil):

ly:text-interface::print

The symbol to print.

vertical-skylines (pair of skylines):

#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 at lily/output-lib.scm:1259:3 (grob)> >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): dynamic-interface (page 771), dynamic-text-interface (page 771), font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), script-interface (page 814), self-alignment-interface (page 815), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.50 DynamicTextSpanner

Dynamic text like ‘cresc’, usually followed by a (dashed) line. See also DynamicLineSpanner (page 595), and TextSpanner (page 723).

DynamicTextSpanner objects are created by the following engraver(s): Dynamic\_engraver (page 470).

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. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 ly:grob::vertical-skylines-from-stencil`

`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 771), `dynamic-text-spanner-interface` (page 771), `font-interface` (page 774), `grob-interface` (page 780), `horizontal-line-spanner-interface` (page 786), `line-interface` (page 794), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.51 Episema

An *episema* line (over a group of notes). Used in Gregorian chant.

Episema objects are created by the following engraver(s): `Episema_engraver` (page 471).

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 ly:side-position-interface::y-aligned-side`

`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `episema-interface` (page 772), `font-interface` (page 774), `grob-interface` (page 780), `horizontal-line-spanner-interface` (page 786), `line-interface` (page 794), `side-position-interface` (page 818), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.52 FingerGlideSpanner

A line connecting two `Fingering` (page 603), `StringNumber` (page 706), or `StrokeFinger` (page 708), grobs, usually indicating a gliding finger, the same string, or the same stroking finger for stringed instruments.

`FingerGlideSpanner` objects are created by the following engraver(s): `Finger_glide_engraver` (page 472).

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))

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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 ly:grob::vertical-skylines-from-stencil
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 772), *font-interface* (page 774), *grob-interface* (page 780), *line-spanner-interface* (page 795), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.53 Fingering

A fingering symbol (usually a digit). See also *FingeringColumn* (page 605), and *StrokeFinger* (page 708).

Fingering objects are created by the following engraver(s): *Fingering\_engraver* (page 472), and *New\_fingering\_engraver* (page 487).

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 not a number, align on the parent’s reference point. 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. If not a number, align on the object’s reference point.

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*.

X-padding (dimension, in staff space):

0.3

Add this much extra space between objects that are next to each other horizontally, overriding the padding property value.

Y-extent (pair of numbers):

#<unpure-pure-container ly:grob::stencil-height >

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

Remarks:

- If the fingering is left- or right-positioned, the Y-offset property is taken relative to the vertical position of its X-parent (i.e., its associated note head).

This object supports the following interface(s): `fingering-interface` (page 773), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `text-interface` (page 837), and `text-script-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.54 FingeringColumn

An auxiliary grob to align stacked Fingering (page 603), grobs.

FingeringColumn objects are created by the following engraver(s): `Fingering_column_engraver` (page 472).

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.

Remarks:

- padding applies to the vertical axis only.

This object supports the following interface(s): `fingering-column-interface` (page 773), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.55 Flag

A flag (in the musical sense).

Flag objects are created by the following engraver(s): `Stem_engraver` (page 500).

Standard settings:

color (color):

#<procedure at lily/output-lib.scm:1722:0 (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 or clefs, *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 at lily/output-lib.scm:1722:0 (grob)>`

This makes the grob invisible.

`vertical-skylines` (pair of skylines):

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 ly:flag::calc-y-offset ly:flag::pure-calc-y-offset >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `flag-interface` (page 774), `font-interface` (page 774), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.56 Footnote

A footnote mark (usually a number) with a pointing line attached to another grob.

Footnote objects are created by the following engraver(s): `Footnote_engraver` (page 473).

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 at lily/output-lib.scm:1661:0 (grob)>  
 If set, footnotes are automatically numbered.

break-visibility (vector):  
 #<procedure at lily/output-lib.scm:3567:0 (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 at lily/output-lib.scm:1661:0 (grob)>  
 A footnote for the grob.

stencil (stencil):  
 ly:balloon-interface::print  
 The symbol to print.

text (markup):  
 #<procedure at lily/output-lib.scm:1661:0 (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 at lily/output-lib.scm:1661:0 (grob)>  
 The horizontal amount that this object is moved relative to its X-parent.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

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 at lily/output-lib.scm:1661:0 (grob)>  
 The vertical amount that this object is moved relative to its Y-parent.



Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): balloon-interface (page 755), font-interface (page 774), footnote-interface (page 776), grob-interface (page 780), sticky-grob-interface (page 833), and text-interface (page 837).

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 790), and spanner-interface (page 826).

### 3.1.57 FretBoard

A fretboard diagram.

FretBoard objects are created by the following engraver(s): Fretboard\_engraver (page 473).

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.
  - barre-thickness – Thickness of barre line, in multiples of dot-radius. Only defined for barre-type=straight. Default value 1.
- 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-overhang` – Extension of string lines beyond last fret line, in multiples of fret-distance. Default value 1.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string  $k$  is given by  $\text{thickness} * (1 + \text{string-thickness-factor})^{(k-1)}$ . Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.

- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`stencil` (`stencil`):

`fret-board::calc-stencil`

The symbol to print.

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 766), `font-interface` (page 774), `fret-diagram-interface` (page 776), `grob-interface` (page 780), `item-interface` (page 790), `outside-staff-interface` (page 808), and `rhythmic-grob-interface` (page 813).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.58 Glissando

A glissando line.

Glissando objects are created by the following engraver(s): `Glissando_engraver` (page 474).

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 ly:grob::vertical-skylines-from-stencil
 ly:grob::pure-simple-vertical-skylines-from-extents >
 Two skylines, one above and one below this grob.

zigzag-width (dimension, in staff space):
 0.75
 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): `font-interface` (page 774), `glissando-interface` (page 778), `grob-interface` (page 780), `line-interface` (page 794), `line-spanner-interface` (page 795), `spanner-interface` (page 826), and `unbreakable-spanner-interface` (page 845).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.59 GraceSpacing

An auxiliary grob to handle (horizontal) spacing of grace notes. See also `NoteSpacing` (page 660), `StaffSpacing` (page 700), and `SpacingSpanner` (page 693).

`GraceSpacing` objects are created by the following engraver(s): `Grace_spacing_engraver` (page 475).

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 778), `grob-interface` (page 780), `spacing-options-interface` (page 824), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.60 GridChordName

A chord name in a chord grid.

GridChordName objects are created by the following engraver(s):  
Grid\_chord\_name\_engraver (page 476).

Standard settings:

font-family (symbol):  
'sans

The font family is the broadest category for selecting text fonts. Options include serif, sans and typewriter.

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 at lily/output-lib.scm:3729:0 (grob)>  
The horizontal amount that this object is moved relative to its X-parent.  
Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-offset (number):  
#<procedure at lily/output-lib.scm:3729:0 (grob)>  
The vertical amount that this object is moved relative to its Y-parent.  
Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): accidental-switch-interface (page 751), font-interface (page 774), grid-chord-name-interface (page 779), grob-interface (page 780), spanner-interface (page 826), and text-interface (page 837).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.61 GridLine

A vertical line between staves, indicating rhythmic synchronization. See also GridPoint (page 613).

GridLine objects are created by the following engraver(s): Grid\_line\_span\_engraver (page 476).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): grid-line-interface (page 779), grob-interface (page 780), item-interface (page 790), and self-alignment-interface (page 815).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.62 GridPoint

An auxiliary grob marking a start or end point for a GridLine (page 612), grob.

GridPoint objects are created by the following engraver(s): Grid\_point\_engraver (page 476).

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 780), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.63 Hairpin

A hairpin. See also `DynamicLineSpanner` (page 595).

Hairpin objects are created by the following engraver(s): `Dynamic_engraver` (page 470).

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 ly:grob::vertical-skylines-from-stencil`

`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 ly:grob::stencil-height 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 ly:self-alignment-interface::y-aligned-on-self ly:self-alignment-interface::pure-y-aligned-on-self >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `dynamic-interface` (page 771), `grob-interface` (page 780), `hairpin-interface` (page 784), `line-interface` (page 794), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.64 HorizontalBracket

A horizontal bracket between notes. See also `HorizontalBracketText` (page 617), and `MeasureSpanner` (page 644).

`HorizontalBracket` objects are created by the following engraver(s): `Horizontal_bracket_engraver` (page 477).

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.



`break-overshoot` (pair of numbers):

`horizontal-bracket::calc-break-overshoot`

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

`connect-to-neighbor` (pair):

`ly: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 UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

`outside-staff-priority` (number):

800

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.

`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`).

`vertical-skylines` (pair of skylines):

`#<unpure-pure-container ly:grob::vertical-skylines-from-stencil  
ly:grob::pure-simple-vertical-skylines-from-extents >`

Two skylines, one above and one below this grob.

Y-offset (number):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): grob-interface (page 780), horizontal-bracket-interface (page 785), line-interface (page 794), outside-staff-interface (page 808), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.65 HorizontalBracketText

Text (markup) for a HorizontalBracket (page 615), grob.

HorizontalBracketText objects are created by the following engraver(s): Horizontal\_bracket\_engraver (page 477).

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 not a number, align on the parent’s reference point. 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. If not a number, align on the object’s reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-offset (number):`

`#<unpure-pure-container ly:side-position-interface::y-aligned-side`

`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `horizontal-bracket-text-interface` (page 786), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.66 InstrumentName

An instrument name, usually displayed to the left of a staff.

`InstrumentName` objects are created by the following engraver(s):

`Instrument_name_engraver` (page 477).

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. If not a number, align on the object's reference point.

`self-alignment-Y (number):`

0

Like `self-alignment-X` but for the Y axis.

stencil (stencil):

system-start-text::print

The symbol to print.

X-offset (number):

system-start-text::calc-x-offset

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-offset (number):

system-start-text::calc-y-offset

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): accidental-switch-interface (page 751), font-interface (page 774), grob-interface (page 780), self-alignment-interface (page 815), side-position-interface (page 818), spanner-interface (page 826), system-start-text-interface (page 835), and text-interface (page 837).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.67 InstrumentSwitch

This grob is deprecated. Do not use it.

InstrumentSwitch objects are created by the following engraver(s):

Instrument\_switch\_engraver (page 478).

Standard settings:

direction (direction):

1

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

extra-spacing-width (pair of numbers):

'(+inf.0 . -inf.0)

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to (+inf.0 . -inf.0).

outside-staff-priority (number):

500

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller outside-staff-priority is closer to the staff.

padding (dimension, in staff space):

0.5

Add this much extra space between objects that are next to each other.

parent-alignment-X (number):

#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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): accidental-switch-interface (page 751), font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.68 JumpScript

A grob to display a ‘point of departure’ like *D.C. al fine*.

JumpScript objects are created by the following engraver(s): `Jump_engraver` (page 478).

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-shape` (symbol):

`'italic`

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

`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):  
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 `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. If not a number, align on the object's reference point.

`stencil` (stencil):  
`ly:text-interface::print`  
The symbol to print.

`vertical-skylines` (pair of skylines):  
`#<unpure-pure-container 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.  
Note that many objects have special positioning considerations, which cause any setting of `X-offset` to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):  
`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side`  
`ly:side-position-interface::pure-y-aligned-side >`  
The vertical amount that this object is moved relative to its Y-parent.  
Note that many objects have special positioning considerations, which cause any setting of `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `break-alignable-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `jump-script-interface` (page 791), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.69 KeyCancellation

A key cancellation, normally consisting of naturals, to be displayed (if necessary) immediately before a `KeySignature` (page 625), grob if the key changes.

KeyCancellation objects are created by the following engraver(s): Key\_engraver (page 479).

Standard settings:

break-align-symbol (symbol):

'key-cancellation

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in *Internals Reference*.

break-visibility (vector):

##(##t ##t ##f)

A vector of 3 booleans, ##(end-of-line unbroken begin-of-line). ##t means visible, ##f means killed.

extra-spacing-height (pair of numbers):

pure-from-neighbor-interface::extra-spacing-height-including-staff

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to (-inf.0 . +inf.0).

extra-spacing-width (pair of numbers):

'(0.0 . 1.0)

In the horizontal spacing problem, we pad each item by this amount (by adding the ‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to (+inf.0 . -inf.0).

flat-positions (list):

'(2 3 4 2 1 2 1)

Flats in key signatures are placed within the specified ranges of staff positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff position at which each clef places C: (alto treble tenor soprano baritone mezzosoprano bass). If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff position.

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):

'((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 shrink-space . 2.5)
(optional-material-end-bracket extra-space . 1.0)
(optional-material-start-bracket
 extra-space
 .
 1.0)
(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)

```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```

extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable 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, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

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 ly:grob::vertical-skylines-from-stencil >`

Two skylines, one above and one below this grob.

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `key-cancellation-interface` (page 791), `key-signature-interface` (page 791), `pure-from-neighbor-interface` (page 812), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.70 KeySignature

A key signature. See also `KeyCancellation` (page 622).

`KeySignature` objects are created by the following engraver(s): `Key_engraver` (page 479).

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)
 (key-cancellation extra-space . 0.3)
 (time-signature extra-space . 1.15)
 (signum-repetitionis extra-space . 1.1)
 (staff-bar extra-space . 1.1)
 (cue-clef extra-space . 0.5)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (right-edge extra-space . 0.5)
 (first-note shrink-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)
```

If space-alist is defined for a grob that gets spaced in a staff, an entry for first-note must be present. If there is no next-note entry, the value of first-note is used instead.

Choices for *spacing-style* are:

```
extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable 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, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

**shrink-space**

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

**semi-shrink-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 shrinkable.

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 ly:grob::vertical-skylines-from-stencil >`

Two skylines, one above and one below this grob.

**Y-extent (pair of numbers):**

`#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `key-signature-interface` (page 791), `pure-from-neighbor-interface` (page 812), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.71 **KievanLigature**

An auxiliary grob to handle a melisma (ligature) as used in Kievan square notation. See also `MensuralLigature` (page 646), `VaticanaLigature` (page 741), and `LigatureBracket` (page 634).

KievanLigature objects are created by the following engraver(s):  
 Kievan\_ligature\_engraver (page 481).

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 774),  
 grob-interface (page 780), kievan-ligature-interface (page 792), and  
 spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.72 LaissezVibrerTie

A laissez-vibrer tie (i.e., a tie from a note into nothing). See also LaissezVibrerTieColumn (page 630), RepeatTie (page 676), and Tie (page 725).

LaissezVibrerTie objects are created by the following engraver(s):  
 Laissez\_vibrer\_engraver (page 481).

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):  
 '((height-limit . 1.0) (ratio . 0.333))

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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?

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*).

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 ly:grob::vertical-skylines-from-stencil >

Two skylines, one above and one below this grob.

Y-extent (pair of numbers):

#<unpure-pure-container 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 762), *grob-interface* (page 780), *item-interface* (page 790), *semi-tie-interface* (page 817), and *tie-interface* (page 838).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.73 LaissezVibrerTieColumn

An auxiliary grob to determine direction and shape of stacked *LaissezVibrerTie* (page 629), grobs.

*LaissezVibrerTieColumn* objects are created by the following engraver(s): *Laissez\_vibrer\_engraver* (page 481).

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 780), `item-interface` (page 790), and `semi-tie-column-interface` (page 816).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.74 LedgerLineSpanner

An auxiliary grob to manage ledger lines of a whole staff.

`LedgerLineSpanner` objects are created by the following engraver(s): `Ledger_line_engraver` (page 481).

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 ly:grob::vertical-skylines-from-stencil`

`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 780), `ledger-line-spanner-interface` (page 793), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).



### 3.1.75 LeftEdge

The left edge of a staff. Useful as an anchor point for other grobs.

LeftEdge objects are created by the following engraver(s): Break\_align\_engraver (page 460).

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)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 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)

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

- extra-space*  
Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.
- shrink-space*  
Only compatible with *first-note* and *next-note*. Put this much space between the two grobs. The space is only shrinkable.
- semi-shrink-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 shrinkable.

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 762), `grob-interface` (page 780), and `item-interface` (page 790).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 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 628), `MensuralLigature` (page 646), and `VaticanaLigature` (page 741).

`LigatureBracket` objects are created by the following engraver(s):  
`Ligature_bracket_engraver` (page 481).

Standard settings:

`bracket-visibility` (boolean or symbol):

#t

This controls the visibility of the tuplet bracket. Setting it to #f 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: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 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*).

`outside-staff-priority` (number):

200

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):

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 780), `line-interface` (page 794), `outside-staff-interface` (page 808), `spanner-interface` (page 826), and `tuplet-bracket-interface` (page 843).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.77 LyricExtender

An extender line in lyrics.

LyricExtender objects are created by the following engraver(s): `Extender_engraver` (page 471).

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*).

This object supports the following interface(s): *grob-interface* (page 780), *lyric-extender-interface* (page 796), *lyric-interface* (page 797), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.78 LyricHyphen

A hyphen in lyrics. See also *VowelTransition* (page 748).

*LyricHyphen* objects are created by the following engraver(s): *Hyphen\_engraver* (page 477).

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 ly:grob::vertical-skylines-from-stencil
 ly:grob::pure-simple-vertical-skylines-from-extents >
 Two skylines, one above and one below this grob.
```

This object supports the following interface(s): *font-interface* (page 774), *grob-interface* (page 780), *lyric-hyphen-interface* (page 797), *lyric-interface* (page 797), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.79 LyricRepeatCount

A repeat count in lyrics.

*LyricRepeatCount* objects are created by the following engraver(s):  
*Lyric\_repeat\_count\_engraver* (page 482).

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):

'normal

Select the series of a font. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

font-shape (symbol):

'italic

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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 at lily/output-lib.scm:1661:0 (grob)>

Text markup. See Section "Formatting text" in *Notation Reference*.

vertical-skylines (pair of skylines):

#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 762), font-interface (page 774), grob-interface (page 780), item-interface (page 790), lyric-interface (page 797), lyric-repeat-count-interface (page 797), self-alignment-interface (page 815), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.80 LyricSpace

A space in lyrics.

LyricSpace objects are created by the following engraver(s): Hyphen\_engraver (page 477).

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 780), lyric-hyphen-interface (page 797), lyric-space-interface (page 798), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).



### 3.1.81 LyricText

A chunk of text in lyrics. See also `LyricExtender` (page 635), `LyricHyphen` (page 636), `LyricSpace` (page 639), and `VowelTransition` (page 748).

`LyricText` objects are created by the following engraver(s): `Lyric_engraver` (page 481).

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):

'normal

Select the series of a font. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

`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 not a number, align on the parent's reference point. If unset, the value from `self-alignment-X` property will be used.

`self-alignment-X` (number):

left-align-at-split-notes

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. If not a number, align on the object's reference point.

`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 at lily/output-lib.scm:1661:0 (grob)>`

Text markup. See Section “Formatting text” in *Notation Reference*.

`vertical-skylines (pair of skylines):`

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 774), `grob-interface` (page 780), `item-interface` (page 790), `lyric-syllable-interface` (page 798), `rhythmic-grob-interface` (page 813), `self-alignment-interface` (page 815), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.82 MeasureCounter

A grob to print a counter for measures.

`MeasureCounter` objects are created by the following engraver(s): `Measure_counter_engraver` (page 484).

Standard settings:

`count-from (integer):`

1

The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`direction (direction):`

1

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

font-encoding (symbol):

'fetaText

The font encoding is the broadest category for selecting a font. Currently, only Lilypond's system fonts (Emmentaler) are using this property. Available values are fetaMusic (Emmentaler), fetaBraces, fetaText (Emmentaler).

font-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 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):

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. If not a number, align on the object's reference point.

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.  
Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-offset` (number):  
#<unpure-pure-container ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >  
The vertical amount that this object is moved relative to its Y-parent.  
Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `centered-spanner-interface` (page 766), `font-interface` (page 774), `grob-interface` (page 780), `measure-counter-interface` (page 798), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.83 MeasureGrouping

A measure grouping or conducting sign.

`MeasureGrouping` objects are created by the following engraver(s):  
`Measure_grouping_engraver` (page 484).

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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): *grob-interface* (page 780), *measure-grouping-interface* (page 798), *outside-staff-interface* (page 808), *side-position-interface* (page 818), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.84 MeasureSpanner

A horizontal bracket between bar lines. See also *HorizontalBracket* (page 615).

*MeasureSpanner* objects are created by the following engraver(s):  
*Measure\_spanner\_engraver* (page 484).

Standard settings:

connect-to-neighbor (pair):

ly: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 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*).

`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. If not a number, align on the object's reference point.

`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):`

'(staff-bar . 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)
```

`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:measure-spanner::print`

The symbol to print.

`Y-offset (number):`

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `line-interface` (page 794), `measure-spanner-interface` (page 799), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 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 the following engraver(s): `Melody_engraver` (page 485).

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 780), `item-interface` (page 790), and `melody-spanner-interface` (page 800).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.86 MensuralLigature

A grob to display a ligature as used in mensural notation. See also `KievanLigature` (page 628), `VaticanaLigature` (page 741), and `LigatureBracket` (page 634).

MensuralLigature objects are created by the following engraver(s): `Mensural_ligature_engraver` (page 485).

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 774), `grob-interface` (page 780), `mensural-ligature-interface` (page 800), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 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 the following engraver(s):  
Metronome\_mark\_engraver (page 485).

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 flags to be displayed within markups (via \note-by-number). Available are 'modern-straight-flag, 'old-straight-flag, 'flat-flag, 'mensural, 'stacked, and 'default.

non-break-align-symbols (list):

'(paper-column-interface)

A list of symbols that determine which NON-break-aligned interfaces to align this to.

outside-staff-horizontal-padding (number):

0.2

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.



`outside-staff-priority` (number):

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. If not a number, align on the object's reference point.

`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 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side`

`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `break-alignable-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `metronome-mark-interface` (page 801), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.88 MultiMeasureRest

A multi-measure rest. See also `MultiMeasureRestNumber` (page 650), `MultiMeasureRestText` (page 653), `MultiMeasureRestScript` (page 652), and `Rest` (page 678).

`MultiMeasureRest` objects are created by the following engraver(s):  
`Multi_measure_rest_engraver` (page 487).

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 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 ly:staff-symbol-referencer::callback >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), multi-measure-interface (page 801), multi-measure-rest-interface (page 801), outside-staff-interface (page 808), rest-interface (page 812), spanner-interface (page 826), and staff-symbol-referencer-interface (page 829).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.89 MultiMeasureRestNumber

A grob to print the length of a MultiMeasureRest (page 649), grob.

MultiMeasureRestNumber objects are created by the following engraver(s): Multi\_measure\_rest\_engraver (page 487).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

side-axis (number):

1

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

springs-and-rods (boolean):

ly:multi-measure-rest::set-text-rods

Dummy variable for triggering spacing routines.

staff-padding (dimension, in staff space):

0.4

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

stencil (stencil):

ly:text-interface::print

The symbol to print.

vertical-skylines (pair of skylines):

#<unpure-pure-container ly:grob::vertical-skylines-from-stencil

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), multi-measure-interface (page 801), multi-measure-rest-number-interface (page 802), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), spanner-interface (page 826), and text-interface (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.90 MultiMeasureRestScript

An articulation (like a fermata) attached to a `MultiMeasureRest` (page 649), `grob`. See also `Script` (page 679).

`MultiMeasureRestScript` objects are created by the following engraver(s): `Multi_measure_rest_engraver` (page 487).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

`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 ly:grob::vertical-skylines-from-stencil

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), multi-measure-interface (page 801), outside-staff-interface (page 808), script-interface (page 814), self-alignment-interface (page 815), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.91 MultiMeasureRestText

A text markup for a MultiMeasureRest (page 649), grob. See also TextScript (page 721).

MultiMeasureRestText objects are created by the following engraver(s):

Multi\_measure\_rest\_engraver (page 487).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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 ly:grob::vertical-skylines-from-stencil
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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), multi-measure-interface (page 801), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), spanner-interface (page 826), and text-interface (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 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 the following engraver(s): `Paper_column_engraver` (page 490).

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 753), font-interface (page 774), grob-interface (page 780), item-interface (page 790), non-musical-paper-column-interface (page 803), paper-column-interface (page 809), separation-item-interface (page 818), and spaceable-grob-interface (page 823).

This object is of class Paper\_column (characterized by paper-column-interface (page 809)).

### 3.1.93 NoteCollision

An auxiliary grob to group NoteColumn (page 657), grobs from several voices, mainly to handle note collisions. See also RestCollision (page 679).

NoteCollision objects are created by the following engraver(s): Collision\_engraver (page 463).

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 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.

This object supports the following interface(s): `axis-group-interface` (page 753), `grob-interface` (page 780), `item-interface` (page 790), and `note-collision-interface` (page 803).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 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 656).

`NoteColumn` objects are created by the following engraver(s): `Rhythmic_column_engraver` (page 495).

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 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.

This object supports the following interface(s): `axis-group-interface` (page 753), `bend-interface` (page 760), `grob-interface` (page 780), `item-interface` (page 790), `note-column-interface` (page 804), and `separation-item-interface` (page 818).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.95 NoteHead

A note head. See also `TabNoteHead` (page 717).

`NoteHead` objects are created by the following engraver(s): `Completion_heads_engraver` (page 463), `Drum_notes_engraver` (page 468), and `Note_heads_engraver` (page 488).

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 or clefs, *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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

`Y-extent` (pair of numbers):  
`#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `accidental-participating-head-interface` (page 750), `bend-interface` (page 760), `font-interface` (page 774), `gregorian-ligature-interface` (page 778), `grob-interface` (page 780), `item-interface` (page 790), `ledgered-interface` (page 793), `ligature-head-interface` (page 794), `mensural-ligature-interface` (page 800), `note-head-interface` (page 805), `rhythmic-grob-interface` (page 813), `rhythmic-head-interface` (page 813), `staff-symbol-referencer-interface` (page 829), and `vaticana-ligature-interface` (page 845).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.96 NoteName

A textual representation of a note name.

`NoteName` objects are created by the following engraver(s): `Note_name_engraver` (page 488).

Standard settings:

`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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 751), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `note-name-interface` (page 806), `self-alignment-interface` (page 815), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.97 NoteSpacing

An auxiliary grob to handle (horizontal) spacing of notes. See also `GraceSpacing` (page 611), `StaffSpacing` (page 700), and `SpacingSpanner` (page 693).

`NoteSpacing` objects are created by the following engraver(s): `Note_spacing_engraver` (page 489).

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 780), `item-interface` (page 790), `note-spacing-interface` (page 806), and `spacing-interface` (page 824).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.98 OptionalMaterialBracket

An in-staff bracket delimiting an optional passage.

`OptionalMaterialBracket` objects are created by the following engraver(s): `Optional_material_bracket_engraver` (page 489).

Standard settings:

`break-align-symbol` (symbol):

`#<procedure at lily/output-lib.scm:1705:0 (grob)>`

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in *Internals Reference*.

`break-visibility` (vector):

`#<procedure at lily/output-lib.scm:1733:0 (grob)>`

A vector of 3 booleans, `#(end-of-line unbroken begin-of-line)`. `#t` means visible, `#f` means killed.

`break-visibility-passage-default` (vector):

`#(#t #t #f)`

The value to use for `break-visibility` when the item does not specifically mark the start or end of a passage. (It might be both or neither, depending on the type of item.)

`break-visibility-passage-end` (vector):

`#(#t #t #f)`

The value to use for `break-visibility` when the item marks the end of a passage.

`break-visibility-passage-start` (vector):

`#(#f #t #t)`

The value to use for `break-visibility` when the item marks the start of a passage.

`direction` (direction):

`#<procedure at lily/output-lib.scm:1761:0 (grob)>`

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`.

positions (pair of numbers):

optional-material-bracket::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):

0.75

In a bracket indicating non-arpeggio or optional material, the length of the horizontal edges.

space-alist (alist, with symbols as keys):

```
'((ambitus extra-space . 2.0)
 (breathing-sign minimum-space . 1.0)
 (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 . 1.0)
 (clef minimum-space . 2.0)
 (cue-clef minimum-space . 2.0)
 (cue-end-clef minimum-space . 2.0)
 (optional-material-end-bracket extra-space . 1.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (first-note extra-space . 0.5)
 (next-note extra-space . 0.5)
 (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)
```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

`extra-space`

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`stencil (stencil):`

`ly:arpeggio::brew-chord-bracket`

The symbol to print.

`thickness (number):`

1.5

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 ly:grob::stencil-height ly:arpeggio::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): `break-aligned-interface` (page 762), `grob-interface` (page 780), `item-interface` (page 790), `optional-material-bracket-interface` (page 807), and `passage-delimiter-interface` (page 810).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.99 `OttavaBracket`

An ottava bracket.

`OttavaBracket` objects are created by the following engraver(s): `Ottava_spanner_engraver` (page 489).

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 . 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. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

`font-shape` (symbol):

'italic

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

`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 ly:grob::vertical-skylines-from-stencil  
ly:grob::pure-simple-vertical-skylines-from-extents >  
Two skylines, one above and one below this grob.

`Y-offset` (number):  
#<unpure-pure-container ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >  
The vertical amount that this object is moved relative to its Y-parent.  
Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `horizontal-bracket-interface` (page 785), `line-interface` (page 794), `ottava-bracket-interface` (page 807), `outside-staff-interface` (page 808), `side-position-interface` (page 818), `spanner-interface` (page 826), and `text-interface` (page 837).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.100 PaperColumn

An auxiliary grob grouping musical items to handle the flexible horizontal space between musical and non-musical columns. See also `NonMusicalPaperColumn` (page 655).

`PaperColumn` objects are created by the following engraver(s): `Paper_column_engraver` (page 490).

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.

`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.

`X-alignment-extent` (pair of numbers):

`'(0 . 1.35)`

If a grob wants to align itself on a `PaperColumn` grob that doesn't contain note heads, use this horizontal extent as a placeholder.

`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 753), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `musical-paper-column-interface` (page 802), `paper-column-interface` (page 809), `separation-item-interface` (page 818), and `spaceable-grob-interface` (page 823).

This object is of class `Paper-column` (characterized by `paper-column-interface` (page 809)).

### 3.1.101 Parentheses

A grob to create parentheses around other grobs.

Parentheses objects are created by the following engraver(s): `Parenthesis_engraver` (page 491).

Standard settings:

`break-visibility` (vector):

`#<procedure at lily/output-lib.scm:3567:0 (grob)>`

A vector of 3 booleans, `#(end-of-line unbroken begin-of-line)`. `#t` means visible, `#f` means killed.

`font-size` (number):

-6

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`padding` (dimension, in staff space):

0.2

Add this much extra space between objects that are next to each other.

`stencil` (stencil):

`parentheses-interface::print`

The symbol to print.

`stencils` (list):

`parentheses-interface::calc-parenthesis-stencils`

Multiple stencils, used as intermediate value.

`Y-extent` (pair of numbers):

`#<unpure-pure-container parentheses-interface::calc-Y-extent >`

Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

`Y-offset` (number):

`#<unpure-pure-container #<procedure at lily/output-lib.scm:1210:0 (grob . rest)>>>`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `parentheses-interface` (page 809), and `sticky-grob-interface` (page 833).

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 790), and `spanner-interface` (page 826).

### 3.1.102 PercentRepeat

A percent symbol for repeating a bar. See also `PercentRepeatCounter` (page 668), `DoublePercentRepeat` (page 590), `DoubleRepeatSlash` (page 592), and `RepeatSlash` (page 675).

`PercentRepeat` objects are created by the following engraver(s): `Percent_repeat_engraver` (page 492).

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. If not a number, align on the object's reference point.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `centered-spanner-interface` (page 766), `font-interface` (page 774), `grob-interface` (page 780), `multi-measure-rest-interface` (page 801), `percent-repeat-interface` (page 810), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.103 PercentRepeatCounter

A grob to print a counter for `PercentRepeat` (page 667), grobs.

PercentRepeatCounter objects are created by the following engraver(s):  
 Percent\_repeat\_engraver (page 492).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side

ly:side-position-interface::pure-y-aligned-side >

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), spanner-interface (page 826), and text-interface (page 837).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.104 PhrasingSlur

A phrasing slur, indicating a ‘musical sentence’. See also Slur (page 688).

PhrasingSlur objects are created by the following engraver(s): Phrasing\_slur\_engraver (page 492).

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):

```
'((absolute-closeness-measure . 0.3)
 (accidental-collision . 3)
 (close-to-edge-length . 2.5)
 (edge-attraction-factor . 4)
 (edge-slope-exponent . 1.7)
 (encompass-object-range-overshoot . 0.5)
 (extra-encompass-collision-distance . 0.8)
 (extra-encompass-free-distance . 0.3)
 (extra-object-collision-penalty . 50)
 (free-head-distance . 0.3)
 (free-slur-distance . 0.8)
 (gap-to-staffline-inside . 0.2)
 (gap-to-staffline-outside . 0.1)
 (head-encompass-penalty . 1000.0))
```

```
(head-slur-distance-factor . 10)
(head-slur-distance-max-ratio . 3)
(max-slope . 1.1)
(max-slope-factor . 10)
(non-horizontal-penalty . 15)
(region-size . 4)
(same-slope-penalty . 20)
(slur-tie-extrema-min-distance . 0.2)
(slur-tie-extrema-min-distance-penalty . 2)
(steeper-slope-factor . 50)
(stem-encompass-penalty . 30.0))
```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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):

```
0.333
```

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):

1.2

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 ly:grob::vertical-skylines-from-stencil
 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 ly:slur::height 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 762), *grob-interface* (page 780), *outside-staff-interface* (page 808), *slur-interface* (page 820), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.105 PianoPedalBracket

A piano pedal bracket. It can also be part of *SostenutoPedal* (page 690), *SustainPedal* (page 710), or *UnaCordaPedal* (page 739), grobs if they are printed in a bracketed style.

*PianoPedalBracket* objects are created by the following engraver(s): *Piano\_pedal\_engraver* (page 492).

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 ly:grob::vertical-skylines-from-stencil
 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 780), *line-interface* (page 794), *piano-pedal-bracket-interface* (page 811), *piano-pedal-interface* (page 811), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.106 RehearsalMark

A rehearsal mark.

RehearsalMark objects are created by the following engraver(s): *Mark\_engraver* (page 482).

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. If not a number, align on the object's reference point.

`stencil (stencil):`

`ly:text-interface::print`

The symbol to print.

`vertical-skylines (pair of skylines):`

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

Y-extent (pair of numbers):

`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-switch-interface` (page 751), `break-alignable-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `mark-interface` (page 798), `outside-staff-interface` (page 808), `rehearsal-mark-interface` (page 812), `self-alignment-interface` (page 815), `side-position-interface` (page 818), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.107 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 667), `DoublePercentRepeat` (page 590), and `DoubleRepeatSlash` (page 592).

`RepeatSlash` objects are created by the following engraver(s): `Slash_repeat_engraver` (page 497).

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 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 780), *item-interface* (page 790), *percent-repeat-interface* (page 810), and *rhythmic-grob-interface* (page 813).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.108 RepeatTie

A repeat tie (i.e., a tie from nothing to a note). See also *RepeatTieColumn* (page 677), *LaissezVibrerTie* (page 629), and *Tie* (page 725).

*RepeatTie* objects are created by the following engraver(s): *Repeat\_tie\_engraver* (page 494).

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):

```
'((height-limit . 1.0) (ratio . 0.333))
```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob’s *details* property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob’s description section.

*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?

`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*).

`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 ly:grob::vertical-skylines-from-stencil >`

Two skylines, one above and one below this grob.

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 762), *grob-interface* (page 780), *item-interface* (page 790), *semi-tie-interface* (page 817), and *tie-interface* (page 838).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.109 RepeatTieColumn

An auxiliary grob to determine direction and shape of stacked *RepeatTie* (page 676), grobs.

*RepeatTieColumn* objects are created by the following engraver(s): *Repeat\_tie\_engraver* (page 494).

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 780), *item-interface* (page 790), and *semi-tie-column-interface* (page 816).

This object is of class *Item* (characterized by *item-interface* (page 790)).

### 3.1.110 Rest

An ordinary rest. See also `MultiMeasureRest` (page 649).

Rest objects are created by the following engraver(s): `Completion_rest_engraver` (page 464), and `Rest_engraver` (page 495).

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 ly:grob::vertical-skylines-from-stencil`

`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 ly:rest::height 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 ly:rest::y-offset-callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `rest-interface` (page 812), `rhythmic-grob-interface` (page 813), `rhythmic-head-interface` (page 813), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.111 RestCollision

An auxiliary grob to handle rest collisions of different voices. See also `NoteCollision` (page 656).

`RestCollision` objects are created by the following engraver(s):  
`Rest_collision_engraver` (page 495).

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 780),  
`item-interface` (page 790), and `rest-collision-interface` (page 812).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.112 Script

An articulation (staccato, accent, etc.). See also `ScriptColumn` (page 680), `ScriptRow` (page 681), and `MultiMeasureRestScript` (page 652).

`Script` objects are created by the following engraver(s): `Drum_notes_engraver` (page 468),  
`New_fingering_engraver` (page 487), and `Script_engraver` (page 496).

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. If not a number, align on the object's reference point.

`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 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.  
Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):  
`#<unpure-pure-container 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 at lily/output-lib.scm:1934:11 (grob . rest)> >`  
The vertical amount that this object is moved relative to its Y-parent.  
Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `horizontal-script-interface` (page 787), `item-interface` (page 790), `outside-staff-interface` (page 808), `script-interface` (page 814), `self-alignment-interface` (page 815), and `side-position-interface` (page 818).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.113 ScriptColumn

An auxiliary grob to (vertically) align stacked Script (page 679), grobs.

ScriptColumn objects are created by the following engraver(s): `Non_musical_script_column_engraver` (page 488), and `Script_column_engraver` (page 495).

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 780), `item-interface` (page 790), and `script-column-interface` (page 813).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.114 ScriptRow

An auxiliary grob to horizontally align stacked Script (page 679), grobs.

ScriptRow objects are created by the following engraver(s): `Script_row_engraver` (page 496).

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 780), `item-interface` (page 790), and `script-column-interface` (page 813).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.115 SectionLabel

A section label, for example ‘Coda’.

SectionLabel objects are created by the following engraver(s): `Mark_engraver` (page 482).

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):  
`'(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. If not a number, align on the object’s reference point.

`stencil (stencil):`

`ly:text-interface::print`

The symbol to print.

`vertical-skylines (pair of skylines):`

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side`

`ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-alignable-interface (page 762), font-interface (page 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), section-label-interface (page 815), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.116 SegnoMark

A segno mark (created with `\repeat segno`, not with `\segno`).

SegnoMark objects are created by the following engraver(s): `Mark_engraver` (page 482).

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)`

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. If not a number, align on the object’s reference point.

`stencil` (stencil):

`ly:text-interface::print`

The symbol to print.

`vertical-skylines` (pair of skylines):

`#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of `X-offset` to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side  
ly:side-position-interface::pure-y-aligned-side >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): break-alignable-interface (page 762), font-interface (page 774), grob-interface (page 780), item-interface (page 790), mark-interface (page 798), outside-staff-interface (page 808), segno-mark-interface (page 815), self-alignment-interface (page 815), side-position-interface (page 818), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.117 SignumRepetitionis

An ancient end-repeat sign.

SignumRepetitionis objects are created by the following engraver(s): Signum\_repetitionis\_engraver (page 497).

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 at lily/output-lib.scm:1761:0 (grob)>
```

The glyph name within the font.

In the context of (span) bar lines or clefs, *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*).

short-bar-extent (pair of numbers):

```
ly:bar-line::calc-short-bar-extent
```

The Y-extent of a short bar line. The default is half the normal bar extent, rounded up to an integer number of staff spaces.

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)
(optional-material-end-bracket extra-space . 1.0)
(optional-material-start-bracket
 extra-space
 .
 1.0)
(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)

```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```

extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable 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, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

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 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 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `pure-from-neighbor-interface` (page 812), and `signum-repetitionis-interface` (page 819).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.118 Slur

A slur. See also `PhrasingSlur` (page 670).

Slur objects are created by the following engraver(s): `Slur_engraver` (page 497).

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):
'((absolute-closeness-measure . 0.3)
 (accidental-collision . 3)
 (close-to-edge-length . 2.5)
 (edge-attraction-factor . 4)
 (edge-slope-exponent . 1.7)
 (encompass-object-range-overshoot . 0.5)
 (extra-encompass-collision-distance . 0.8)
 (extra-encompass-free-distance . 0.3)
 (extra-object-collision-penalty . 50)
 (free-head-distance . 0.3)
 (free-slur-distance . 0.8)
 (gap-to-staffline-inside . 0.2)
 (gap-to-staffline-outside . 0.1)
 (head-encompass-penalty . 1000.0)
 (head-slur-distance-factor . 10)
 (head-slur-distance-max-ratio . 3)
 (max-slope . 1.1)
 (max-slope-factor . 10)
 (non-horizontal-penalty . 15)
 (region-size . 4)
 (same-slope-penalty . 20)
 (slur-tie-extrema-min-distance . 0.2)
 (slur-tie-extrema-min-distance-penalty . 2)
 (steeper-slope-factor . 50)
 (stem-encompass-penalty . 30.0))

```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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.

font-size (number):

```
-6
```

The font size, compared to the 'normal' size. 0 is style-sheet's normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

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 ly:grob::vertical-skylines-from-stencil  
 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 ly:slur::height 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 762), *grob-interface* (page 780), *outside-staff-interface* (page 808), *slur-interface* (page 820), and *spanner-interface* (page 826).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.119 SostenutoPedal

A *sostenuto* pedal mark. See also *SostenutoPedalLineSpanner* (page 692), *PianoPedalBracket* (page 672), *SustainPedal* (page 710), and *UnaCordaPedal* (page 739).

*SostenutoPedal* objects are created by the following engraver(s): *Piano\_pedal\_engraver* (page 492).

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. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

stencil (stencil):

ly:text-interface::print

The symbol to print.

vertical-skylines (pair of skylines):

#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container 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 774), `grob-interface` (page 780), `item-interface` (page 790), `piano-pedal-script-interface` (page 811), `self-alignment-interface` (page 815), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.120 `SostenutoPedalLineSpanner`

An auxiliary grob providing a baseline to align consecutive `SostenutoPedal` (page 690), grobs vertically.

`SostenutoPedalLineSpanner` objects are created by the following engraver(s): `Piano_pedal_align_engraver` (page 492).

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 ly:grob::vertical-skylines-from-element-stencils  
 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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
```

```
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), grob-interface (page 780), outside-staff-interface (page 808), piano-pedal-interface (page 811), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.121 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 611), `NoteSpacing` (page 660), and `StaffSpacing` (page 700).

`SpacingSpanner` objects are created by the following engraver(s): `Spacing_engraver` (page 498).

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 780), spacing-options-interface (page 824), spacing-spanner-interface (page 824), and spanner-interface (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.122 SpanBar

A span bar, i.e., the parts of a multi-staff bar line that are outside of staves. See also `SpanBarStub` (page 695).

`SpanBar` objects are created by the following engraver(s): `Span_bar_engraver` (page 498).

Standard settings:

allow-span-bar (boolean):  
   #t  
 If false, no inter-staff bar line will be created below this bar line.

bar-extent (pair of numbers):  
   #<unpure-pure-container ly:axis-group-interface::height  
   ly:axis-group-interface::pure-height >

The Y-extent of the actual bar line. This may differ from Y-extent because it does not include the dots in a repeat bar line.

before-line-breaking (boolean):  
   ly:span-bar::before-line-breaking  
 Dummy property, used to trigger a callback function.

break-align-anchor (number):  
   ly:span-bar::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*.

glyph-name (string):  
   ly:span-bar::calc-glyph-name  
 The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

layer (integer):  
   0

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

```

non-musical (boolean):
 #t
 True if the grob belongs to a NonMusicalPaperColumn.

stencil (stencil):
 ly:span-bar::print
 The symbol to print.

X-extent (pair of numbers):
 ly:span-bar::width
 Extent (size) in the X direction, measured in staff-space units, relative to object's
 reference point.

Y-extent (pair of numbers):
 '(+inf.0 . -inf.0)
 Extent (size) in the Y direction, measured in staff-space units, relative to object's
 reference point.

```

This object supports the following interface(s): `bar-line-interface` (page 755), `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), and `span-bar-interface` (page 825).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.123 SpanBarStub

An auxiliary grob, acting like a fake `SpanBar` (page 694), grob in contexts such as `Lyrics` (page 218), that are crossed by a span bar, to keep span bars taking horizontal space.

`SpanBarStub` objects are created by the following engraver(s): `Span_bar_stub_engraver` (page 498).

Standard settings:

```

allow-span-bar (boolean):
 #t
 If false, no inter-staff bar line will be created below this bar line.

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 at lily/output-lib.scm:1722:0 (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 pure-from-neighbor-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): `grob-interface` (page 780), `item-interface` (page 790), `pure-from-neighbor-interface` (page 812), and `span-bar-stub-interface` (page 826).

This object is of class `Item` (characterized by `item-interface` (page 790)).



### 3.1.124 StaffEllipsis

A visual marker (usually three consecutive dots) to indicate that typesetting of music is skipped.

StaffEllipsis objects are created by the following engraver(s):

Skip\_typesetting\_engraver (page 497).

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):

#<procedure at lily/output-lib.scm:1733:0 (grob)>

A vector of 3 booleans, #(end-of-line unbroken begin-of-line). #t means visible, #f means killed.

break-visibility-passage-default (vector):

#(#t #t #f)

The value to use for break-visibility when the item does not specifically mark the start or end of a passage. (It might be both or neither, depending on the type of item.)

break-visibility-passage-end (vector):

#(#t #t #f)

The value to use for break-visibility when the item marks the end of a passage.

break-visibility-passage-start (vector):

#(#f #t #t)

The value to use for break-visibility when the item marks the start of a passage.

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)
 (left-edge extra-space . 0.0)
 (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)
 (optional-material-end-bracket extra-space . 1.0))
```

```
(optional-material-start-bracket
 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)
```

If *space-alist* is defined for a grob that gets spaced in a staff, an entry for *first-note* must be present. If there is no *next-note* entry, the value of *first-note* is used instead.

Choices for *spacing-style* are:

```
extra-space
 Put this much space between the two grobs. The space is stretchable
 and shrinkable 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, with-
 out allowing them to collide. The space is stretchable and shrinkable
 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 and shrinkable.
```

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

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 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. If `whiteout-color` is set, use this color instead of white for the background.

`Y-extent (pair of numbers):`

```
#<unpure-pure-container ly:grob::stencil-height staff-ellipsis::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): `break-aligned-interface` (page 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `passage-delimiter-interface` (page 810), `staff-ellipsis-interface` (page 827), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.125 StaffGrouper

An auxiliary grob to manage vertical spacing of staff groups. See also `VerticalAlignment` (page 742), and `VerticalAxisGroup` (page 742).

StaffGrouper objects are created by the following engraver(s):  
 Vertical\_align\_engraver (page 507).

Standard settings:

```
staff-staff-spacing (alist, with symbols as keys):
 '((basic-distance . 9)
 (minimum-distance . 7)
 (padding . 1)
 (stretchability . 5))
```

When applied to a staff-group's StaffGrouper grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's VerticalAxisGroup grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the StaffGrouper grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

- `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

```
staffgroup-staff-spacing (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 780), `spanner-interface` (page 826), and `staff-grouper-interface` (page 827).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.126 StaffHighlight

A colored span to highlight a music passage.

StaffHighlight objects are created by the following engraver(s):  
 Staff\_highlight\_engraver (page 499).

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):`

`#<procedure at lily/output-lib.scm:1661:0 (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 780), `spanner-interface` (page 826), and `staff-highlight-interface` (page 828).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.127 StaffSpacing

An auxiliary grob to handle spacing within a staff. See also `NoteSpacing` (page 660), `GraceSpacing` (page 611), and `SpacingSpanner` (page 693).

`StaffSpacing` objects are created by the following engraver(s): `Separating_line_group_engraver` (page 496).

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 780), `item-interface` (page 790), `spacing-interface` (page 824), and `staff-spacing-interface` (page 828).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.128 StaffSymbol

A staff symbol, usually five horizontal lines.

`StaffSymbol` objects are created by the following engraver(s): `Staff_symbol_engraver` (page 500), and `Tab_staff_symbol_engraver` (page 502).

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.

`line-positions` (list):

`ly:staff-symbol::calc-line-positions`

Vertical positions 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 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 780), `spanner-interface` (page 826), and `staff-symbol-interface` (page 828).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.129 StanzaNumber

A stanza number (or markup) for lyrics.

StanzaNumber objects are created by the following engraver(s): `Stanza_number_engraver` (page 500).

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. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`

`#<unpure-pure-container 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 774), `grob-interface` (page 780), `item-interface` (page 790), `side-position-interface` (page 818), `stanza-number-interface` (page 830), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.130 Stem

A stem. See also StemStub (page 705).

Stem objects are created by the following engraver(s): Span\_stem\_engraver (page 499), and Stem\_engraver (page 500).

Standard settings:

beamlet-default-length (pair):

'(1.1 . 1.1)

A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by beamlet-max-length-proportion, whichever is smaller.

beamlet-max-length-proportion (pair):

'(0.75 . 0.75)

The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

default-direction (direction):

ly:stem::calc-default-direction

Direction determined by note head positions.

details (alist, with symbols as keys):

'((beamed-extreme-minimum-free-lengths 2.0 1.25)  
(beamed-lengths 3.26 3.5 3.6)  
(beamed-minimum-free-lengths 1.83 1.5 1.25)  
(lengths 3.5 3.5 3.5 4.25 5.0 6.0 7.0 8.0 9.0)  
(stem-shorten 1.0 0.5 0.25))

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

direction (direction):

ly:stem::calc-direction

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

double-stem-separation (number):

0.5

The distance between the two stems of a half note in tablature when using \tabFullNotation, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

duration-log (integer):

stem::calc-duration-log

The 2-log of the note head duration, i.e., 0 = whole note, 1 = half note, etc.

length (dimension, in staff space):

#<unpure-pure-container ly:stem::calc-length 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 ly:stem::calc-stem-begin-position  
 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.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

`Y-extent` (pair of numbers):  
 #<unpure-pure-container 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):  
 #<unpure-pure-container ly:staff-symbol-referencer::callback >  
 The vertical amount that this object is moved relative to its Y-parent.  
 Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `grob-interface` (page 780), `item-interface` (page 790), and `stem-interface` (page 830).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.131 StemStub

An auxiliary grob that prevents cross-staff Stem (page 703), grobs from colliding with articulations.

StemStub objects are created by the following engraver(s): Stem\_engraver (page 500).

Standard settings:

extra-spacing-height (pair of numbers):

stem-stub::extra-spacing-height

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

X-extent (pair of numbers):

stem-stub::width

Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.

Y-extent (pair of numbers):

#<unpure-pure-container #f stem-stub::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): grob-interface (page 780), and item-interface (page 790).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.132 StemTremolo

A stem tremolo.

StemTremolo objects are created by the following engraver(s): Stem\_engraver (page 500).

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 not a number,

align on the parent's reference point. 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.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent (pair of numbers):`  
`#<unpure-pure-container ly:grob::stencil-height ly:stem-tremolo::pure-height >`  
 Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset (number):`  
`#<unpure-pure-container ly:stem-tremolo::calc-y-offset ly:stem-tremolo::pure-calc-y-offset >`  
 The vertical amount that this object is moved relative to its Y-parent.  
 Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `grob-interface` (page 780), `item-interface` (page 790), `self-alignment-interface` (page 815), and `stem-tremolo-interface` (page 832).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.133 StringNumber

A markup (by default a digit in a circle) to name a string.

`StringNumber` objects are created by the following engraver(s): `New_fingering_engraver` (page 487).

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 not a number, align on the parent's reference point. If unset, the value from `self-alignment-X` property will be used.

`script-priority (number):`

`150`

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. If not a number, align on the object's reference point.

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 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 774), grob-interface (page 780), item-interface (page 790), number-interface (page 806), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), string-number-interface (page 833), text-interface (page 837), and text-script-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.134 StrokeFinger

A markup (usually a lowercase letter) to indicate right-hand fingering. See also Fingering (page 603).

StrokeFinger objects are created by the following engraver(s): New\_fingering\_engraver (page 487).

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.

digit-names (vector):

#("p" "i" "m" "a" "x")

Names for string finger digits.

font-shape (symbol):

'italic

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

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 not a number, align on the parent's reference point. If unset, the value from `self-alignment-X` property will be used.

script-priority (number):

125

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. If not a number, align on the object's reference point.

self-alignment-Y (number):

0

Like `self-alignment-X` but for the Y axis.

slur-padding (number):

0.3

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):

stroke-finger::calc-text

Text markup. See Section "Formatting text" in *Notation Reference*.

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 774), grob-interface (page 780), item-interface (page 790), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), stroke-finger-interface (page 833), text-interface (page 837), and text-script-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.135 SustainPedal

A sustain pedal mark. See also SustainPedalLineSpanner (page 711), PianoPedalBracket (page 672), SostenutoPedal (page 690), and UnaCordaPedal (page 739).

SustainPedal objects are created by the following engraver(s): Piano\_pedal\_engraver (page 492).

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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

stencil (stencil):

```
ly:sustain-pedal::print
```

The symbol to print.

vertical-skylines (pair of skylines):

```
#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 774), grob-interface (page 780), item-interface (page 790), piano-pedal-interface (page 811), piano-pedal-script-interface (page 811), self-alignment-interface (page 815), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.136 SustainPedalLineSpanner

An auxiliary grob providing a baseline to align consecutive SustainPedal (page 710), grobs vertically.

SustainPedalLineSpanner objects are created by the following engraver(s): Piano\_pedal\_align\_engraver (page 492).

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 ly:grob::vertical-skylines-from-element-stencils
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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), grob-interface (page 780), outside-staff-interface (page 808), piano-pedal-interface (page 811), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.137 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 ly:system::height 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 753), `grob-interface` (page 780), `outside-staff-axis-group-interface` (page 807), `spanner-interface` (page 826), and `system-interface` (page 833).

This object is of class `System` (characterized by `system-interface` (page 833)).

### 3.1.138 SystemStartBar

A bar line as a system start delimiter.

`SystemStartBar` objects are created by the following engraver(s):

`System_start_delimiter_engraver` (page 501).

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.  
 Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): *grob-interface* (page 780), *side-position-interface* (page 818), *spanner-interface* (page 826), and *system-start-delimiter-interface* (page 834).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.139 SystemStartBrace

A brace as a system start delimiter.

*SystemStartBrace* objects are created by the following engraver(s):  
*System\_start\_delimiter\_engraver* (page 501).

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.  
Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), side-position-interface (page 818), spanner-interface (page 826), and system-start-delimiter-interface (page 834).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.140 SystemStartBracket

A bracket as a system start delimiter.

`SystemStartBracket` objects are created by the following engraver(s):  
`System_start_delimiter_engraver` (page 501).

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): *font-interface* (page 774), *grob-interface* (page 780), *side-position-interface* (page 818), *spanner-interface* (page 826), and *system-start-delimiter-interface* (page 834).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.141 SystemStartSquare

A rectangle-like bracket as a start delimiter.

*SystemStartSquare* objects are created by the following engraver(s): *System\_start\_delimiter\_engraver* (page 501).

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), side-position-interface (page 818), spanner-interface (page 826), and system-start-delimiter-interface (page 834).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.142 TabNoteHead

A ‘note head’ (usually a digit) in a tablature. See also NoteHead (page 658).

TabNoteHead objects are created by the following engraver(s): Tab\_note\_heads\_engraver (page 501).

Standard settings:

after-line-breaking (boolean):

tab-note-head::handle-ties

Dummy property, used to trigger callback for after-line-breaking.

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))

(harmonic-properties

(angularity . 2)

(half-thickness . 0.075)

(padding . 0)

(procedure

.

#<procedure parenthesize-stencil (stencil half-thickness width angularity padding  
(width . 0.25))

(head-offset . 3/5)

(tied-properties

(note-head-visible . #t)

(parenthesize . #t)

(repeat-tied . #f)

(tied . #f)))

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob’s details property. See Section 3.2 [Graphical Object Interfaces],

page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

`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. If `whiteout-color` is set, use this color instead of white for the background.

X-offset (number):

`ly:self-alignment-interface::x-aligned-on-self`

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

Y-extent (pair of numbers):

`#<unpure-pure-container 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `bend-interface` (page 760), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `note-head-interface` (page 805), `rhythmic-grob-interface` (page 813), `rhythmic-head-interface` (page 813), `staff-symbol-referencer-interface` (page 829), `tab-note-head-interface` (page 835), and `text-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.143 TextMark

An arbitrary textual mark. See also `SectionLabel` (page 681), and `JumpScript` (page 621), for grobs with a more specific intent.

`TextMark` objects are created by the following engraver(s): `Text_mark_engraver` (page 503).

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):

`text-mark-interface::calc-break-visibility`

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):`

0.5

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):`

1250

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):`

`text-mark-interface::calc-self-alignment-X`

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. If not a number, align on the object's reference point.

`stencil (stencil):`

`ly:text-interface::print`

The symbol to print.

`text (markup):`

`#<procedure at lily/output-lib.scm:1661:0 (grob)>`

Text markup. See Section "Formatting text" in *Notation Reference*.

vertical-skylines (pair of skylines):

```
#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): accidental-switch-interface (page 751), break-alignable-interface (page 762), font-interface (page 774), grob-interface (page 780), item-interface (page 790), mark-interface (page 798), outside-staff-interface (page 808), self-alignment-interface (page 815), side-position-interface (page 818), text-interface (page 837), and text-mark-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.144 TextScript

A markup attached to a grob like a note head. See also MultiMeasureRestText (page 653).

TextScript objects are created by the following engraver(s): Text\_engraver (page 503).

Standard settings:

avoid-slur (symbol):

```
'around
```

Method of handling slur collisions. Choices are inside, outside, around, and ignore. inside adjusts the slur if needed to keep the grob inside the slur. outside moves the grob vertically to the outside of the slur. around moves the grob vertically to the outside of the slur only if there is a collision. ignore does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), outside and around behave like ignore.

direction (direction):

```
-1
```

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

extra-spacing-width (pair of numbers):

'(+inf.0 . -inf.0)

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to (+inf.0 . -inf.0).

outside-staff-horizontal-padding (number):

0.2

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

outside-staff-priority (number):

450

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller outside-staff-priority is closer to the staff.

padding (dimension, in staff space):

0.3

Add this much extra space between objects that are next to each other.

parent-alignment-X (number):

#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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

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 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.
 Note that many objects have special positioning considerations, which cause any
 setting of X-offset to be ignored or modified, even though the object supports the
 self-alignment-interface (page 815).

Y-extent (pair of numbers):
 #<unpure-pure-container 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 ly:side-position-interface::y-aligned-side
 ly:side-position-interface::pure-y-aligned-side >
 The vertical amount that this object is moved relative to its Y-parent.
 Note that many objects have special positioning considerations, which cause any
 setting of Y-offset to be ignored or modified, even though the object supports the
 self-alignment-interface (page 815).

```

This object supports the following interface(s): `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `instrument-specific-markup-interface` (page 788), `item-interface` (page 790), `outside-staff-interface` (page 808), `self-alignment-interface` (page 815), `side-position-interface` (page 818), `text-interface` (page 837), and `text-script-interface` (page 837).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.145 TextSpanner

Text like ‘rit’, usually followed by a (dashed) line. See also `DynamicTextSpanner` (page 599).

`TextSpanner` objects are created by the following engraver(s): `Text_spanner_engraver` (page 503).

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. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

`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 ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): font-interface (page 774), grob-interface (page 780), horizontal-line-spanner-interface (page 786), line-interface (page 794), outside-staff-interface (page 808), side-position-interface (page 818), and spanner-interface (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.146 Tie

A tie. See also `TieColumn` (page 727), `LaissezVibrerTie` (page 629), and `RepeatTie` (page 676).

Tie objects are created by the following engraver(s): `Completion_heads_engraver` (page 463), and `Tie_engraver` (page 503).

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):

```
'((between-length-limit . 1.0)
 (center-staff-line-clearance . 0.6)
 (dot-collision-clearance . 0.25)
 (dot-collision-penalty . 0.25)
 (height-limit . 1.0)
 (horizontal-distance-penalty-factor . 10)
 (intra-space-threshold . 1.25)
 (min-length . 1.0)
 (min-length-penalty-factor . 26)
 (multi-tie-region-size . 3)
 (note-head-gap . 0.2)
 (outer-tie-length-symmetry-penalty-factor . 10)
 (outer-tie-vertical-distance-symmetry-penalty-factor
 .
 10)
 (outer-tie-vertical-gap . 0.25)
 (ratio . 0.333))
```

```
(same-dir-as-stem-penalty . 8)
(single-tie-region-size . 4)
(skyline-padding . 0.05)
(staff-line-collision-penalty . 5)
(stem-gap . 0.35)
(tie-column-monotonicity-penalty . 100)
(tie-tie-collision-distance . 0.45)
(tie-tie-collision-penalty . 25.0)
(tip-staff-line-clearance . 0.45)
(vertical-distance-penalty-factor . 7)
(wrong-direction-offset-penalty . 10))
```

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob’s details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob’s description section.

**direction** (direction):

ly:tie::calc-direction

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

**font-size** (number):

-6

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

**line-thickness** (number):

0.8

For slurs and ties, this is the diameter of the virtual “pen” that draws the two arcs of the curve’s outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

**neutral-direction** (direction):

1

Which direction to take in the center of the staff.

**springs-and-rods** (boolean):

ly:spanner::set-spacing-rods

Dummy variable for triggering spacing routines.

**stencil** (stencil):

ly:tie::print

The symbol to print.

**thickness** (number):

1.2

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve’s outline at its thickest point, not

counting the diameter of the virtual “pen” that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

*vertical-skylines* (pair of skylines):

```
#<unpure-pure-container ly:grob::vertical-skylines-from-stencil
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 762), *grob-interface* (page 780), *spanner-interface* (page 826), and *tie-interface* (page 838).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.147 TieColumn

An auxiliary grob to determine direction and shape of stacked Tie (page 725), grobs.

TieColumn objects are created by the following engraver(s): *Completion\_heads\_engraver* (page 463), and *Tie\_engraver* (page 503).

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 780), *spanner-interface* (page 826), and *tie-column-interface* (page 838).

This object is of class *Spanner* (characterized by *spanner-interface* (page 826)).

### 3.1.148 TimeSignature

A time signature.

TimeSignature objects are created by the following engraver(s): *Time\_signature\_engraver* (page 504).

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.

`denominator-style` (symbol):

'default

The style of denominators in a time signature.

`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)`.

`nested-fraction-mixed` (boolean):

`##t`

Whether a fractional term of a time signature is printed as a mixed number (e.g., '2 1/2') or as a common fraction (e.g., '5/2').

`nested-fraction-orientation` (symbol):

'default

A symbol describing the orientation of a fractional part of a time signature.

`non-musical` (boolean):

`##t`

True if the grob belongs to a `NonMusicalPaperColumn`.

`note-dots-direction` (direction):

0

Whether the augmentation dots are shifted up or down (or not shifted) relative to the note head in a number-over-note time signature.

`note-staff-position` (number):

-1

The position of the note in a number-over-note time signature. See `staff-position`.

`senza-misura-stencil` (stencil):

#f

The symbol to print when `TimeSignature.fraction` is not set. Overriding `TimeSignature.stencil` circumvents this.

`space-alist` (alist, with symbols as keys):

```
'((ambitus extra-space . 1.0)
 (cue-clef extra-space . 1.5)
 (custos minimum-space . 0.5)
 (first-note semi-shrink-space . 2.0)
 (optional-material-start-bracket
 extra-space
 .
 1.0)
 (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)

If `space-alist` is defined for a grob that gets spaced in a staff, an entry for `first-note` must be present. If there is no `next-note` entry, the value of `first-note` is used instead.

Choices for *spacing-style* are:

`extra-space`

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable

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 and shrinkable.

#### `shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

#### `semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

#### `stencil (stencil):`

`ly:time-signature::print`

The symbol to print.

#### `style (symbol):`

`'C`

This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

#### `Y-extent (pair of numbers):`

`#<unpure-pure-container 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 762), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `pure-from-neighbor-interface` (page 812), and `time-signature-interface` (page 841).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.149 TrillPitchAccidental

The accidental of a pitched trill. See also `TrillPitchGroup` (page 731).

`TrillPitchAccidental` objects are created by the following engraver(s): `Pitched_trill_engraver` (page 494).

Standard settings:

`direction (direction):`

-1

If `side-axis` is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

`font-size` (number):

-4

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`glyph-name` (string):

`accidental-interface::calc-glyph-name`

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`Y-extent` (pair of numbers):

`#<unpure-pure-container 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 749), `accidental-switch-interface` (page 751), `font-interface` (page 774), `grob-interface` (page 780), `inline-accidental-interface` (page 787), `item-interface` (page 790), `side-position-interface` (page 818), and `trill-pitch-accidental-interface` (page 842).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.150 TrillPitchGroup

An auxiliary grob to construct a pitched trill, aligning `TrillPitchAccidental` (page 730), `TrillPitchParentheses` (page 733), and `TrillPitchHead` (page 733), horizontally. See also `TrillSpanner` (page 734).

TrillPitchGroup objects are created by the following engraver(s):  
 Pitched\_trill\_engraver (page 494).

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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

#<unpure-pure-container ly:axis-group-interface::height  
 trill-pitch-group::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 753), grob-interface (page 780), item-interface (page 790), and side-position-interface (page 818).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.151 TrillPitchHead

The note head of a pitched trill. See also `TrillPitchGroup` (page 731).

`TrillPitchHead` objects are created by the following engraver(s):  
`Pitched_trill_engraver` (page 494).

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 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 ly:staff-symbol-referencer::callback >`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `accidental-participating-head-interface` (page 750), `font-interface` (page 774), `grob-interface` (page 780), `item-interface` (page 790), `ledgered-interface` (page 793), `note-head-interface` (page 805), `pitched-trill-interface` (page 811), and `staff-symbol-referencer-interface` (page 829).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.152 TrillPitchParentheses

The parentheses of a pitched trill. See also `TrillPitchGroup` (page 731).

`TrillPitchParentheses` objects are created by the following engraver(s):  
`Pitched_trill_engraver` (page 494).

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):

`#<unpure-pure-container 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 774), `grob-interface` (page 780), `item-interface` (page 790), `parentheses-interface` (page 809), and `pitched-trill-interface` (page 811).

This object is of class `Item` (characterized by `item-interface` (page 790)).

### 3.1.153 TrillSpanner

A continued trill with a wiggly line (created with `\startTrillSpan`, not with `\trill`). See also `TrillPitchGroup` (page 731).

`TrillSpanner` objects are created by the following engraver(s): `Trill_spanner_engraver` (page 506).

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-dimension-from-markup (layout props axis arg1 arg2)>
0
 (#<procedure with-true-dimension-markup (layout props axis arg)>
0
 (#<procedure musicglyph-markup (layout props glyph-name)>
 "scripts.trill"))
 (#<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.

`vertical-skylines (pair of skylines):`

```
#<unpure-pure-container ly:grob::vertical-skylines-from-stencil
ly:grob::pure-simple-vertical-skylines-from-extents >
```

Two skylines, one above and one below this grob.

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `horizontal-line-spanner-interface` (page 786), `line-interface` (page 794), `outside-staff-interface` (page 808), `side-position-interface` (page 818), `spanner-interface` (page 826), and `trill-spanner-interface` (page 842).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).



### 3.1.154 TupletBracket

A tuplet bracket. See also `TupletNumber` (page 737).

`TupletBracket` objects are created by the following engraver(s): `Tuplet_engraver` (page 506).

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:spanner::calc-connect-to-neighbors`

Pair of booleans, indicating whether this grob looks as a continued break.

`direction` (direction):

`ly:tuplet-bracket::calc-direction`

If `side-axis` is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

`edge-height` (pair):

`'(0.7 . 0.7)`

A pair of numbers specifying the heights of the vertical edges: (*left-height* . *right-height*).

`full-length-to-extent` (boolean):

`#t`

Run to the extent of the column for a full-length tuplet bracket.

`max-slope-factor` (non-negative number):

0.5

Factor for calculating the maximum tuplet bracket slope. Notice that there exists a homonymous property for slurs.

`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.

`span-all-note-heads` (boolean):

`#f`

If true, tuplet brackets are printed spanning horizontally from the first to the last note head instead of covering only the stems.

`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 ly:grob::vertical-skylines-from-stencil  
ly:grob::pure-simple-vertical-skylines-from-extents >  
Two skylines, one above and one below this grob.

`visible-over-note-heads` (boolean):  
#f  
This prints a tuplet bracket when the bracket is set to be over the note heads. This option can be combined with the default tuplet bracket visibility style and with #'if-no-beam.

`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 780), `line-interface` (page 794), `outside-staff-interface` (page 808), `spanner-interface` (page 826), and `tuplet-bracket-interface` (page 843).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.155 TupletNumber

A tuplet number. See also `TupletBracket` (page 736).

`TupletNumber` objects are created by the following engraver(s): `Tuplet_engraver` (page 506).

Standard settings:

`avoid-slur` (symbol):  
'inside

Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs

whose notational significance depends on vertical position (such as accidentals, clefs, etc.), outside and around behave like ignore.

direction (direction):

`tuplet-number::calc-direction`

If `side-axis` is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

font-shape (symbol):

`'italic`

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

font-size (number):

`-2`

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

knee-to-beam (boolean):

`#t`

Determines whether a tuplet number will be positioned next to a kneed beam.

stencil (stencil):

`ly:tuplet-number::print`

The symbol to print.

text (markup):

`tuplet-number::calc-denominator-text`

Text markup. See Section “Formatting text” in *Notation Reference*.

X-offset (number):

`ly:tuplet-number::calc-x-offset`

The horizontal amount that this object is moved relative to its X-parent.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

Y-offset (number):

`ly:tuplet-number::calc-y-offset`

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `font-interface` (page 774), `grob-interface` (page 780), `outside-staff-interface` (page 808), `spanner-interface` (page 826), `text-interface` (page 837), and `tuplet-number-interface` (page 845).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.156 UnaCordaPedal

An una corda pedal mark. See also `UnaCordaPedalLineSpanner` (page 740), `SostenutoPedal` (page 690), `SustainPedal` (page 710), and `PianoPedalBracket` (page 672).

`UnaCordaPedal` objects are created by the following engraver(s): `Piano_pedal_engraver` (page 492).

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. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

`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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

`stencil` (`stencil`):

ly:text-interface::print

The symbol to print.

`vertical-skylines` (pair of skylines):

#<unpure-pure-container 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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

Y-extent (pair of numbers):

```
#<unpure-pure-container 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 774), grob-interface (page 780), item-interface (page 790), piano-pedal-script-interface (page 811), self-alignment-interface (page 815), and text-interface (page 837).

This object is of class Item (characterized by item-interface (page 790)).

### 3.1.157 UnaCordaPedalLineSpanner

An auxiliary grob providing a baseline to align consecutive UnaCordaPedal (page 739), grobs vertically.

UnaCordaPedalLineSpanner objects are created by the following engraver(s): Piano\_pedal\_align\_engraver (page 492).

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 ly:grob::vertical-skylines-from-element-stencils
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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): `axis-group-interface` (page 753), `grob-interface` (page 780), `outside-staff-interface` (page 808), `piano-pedal-interface` (page 811), `side-position-interface` (page 818), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.158 VaticanaLigature

A grob to display a melisma (ligature) as used in Gregorian chant. See also `KievanLigature` (page 628), `MensuralLigature` (page 646), and `LigatureBracket` (page 634).

`VaticanaLigature` objects are created by the following engraver(s):

`Vaticana_ligature_engraver` (page 507).

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 774), `grob-interface` (page 780), `spanner-interface` (page 826), and `vaticana-ligature-interface` (page 845).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.159 VerticalAlignment

A top-level auxiliary grob to stack groups (staves, lyrics lines, etc.). See also `StaffGrouper` (page 698), and `VerticalAxisGroup` (page 742).

`VerticalAlignment` objects are created by the following engraver(s): `Vertical_align_engraver` (page 507).

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 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.

This object supports the following interface(s): `align-interface` (page 751), `axis-group-interface` (page 753), `grob-interface` (page 780), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.160 VerticalAxisGroup

An auxiliary grob to group everything contained in a context like `Staff` (page 309), `Lyrics` (page 218), `Dynamics` (page 130), etc. See also `StaffGrouper` (page 698), and `VerticalAlignment` (page 742).

`VerticalAxisGroup` objects are created by the following engraver(s): `Axis_group_engraver` (page 453).

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 ly:axis-group-interface::calc-staff-staff-spacing
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 ly:hara-kiri-group-spanner::y-extent`

`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.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `axis-group-interface` (page 753), `grob-interface` (page 780), `hara-kiri-group-spanner-interface` (page 785), `outside-staff-axis-group-interface` (page 807), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.161 VoiceFollower

A line to indicate staff changes of a voice.

VoiceFollower objects are created by the following engraver(s):  
`Note_head_line_engraver` (page 488).

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 780), `line-interface` (page 794), `line-spanner-interface` (page 795), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.162 VoltaBracket

A volta bracket. See also `VoltaBracketSpanner` (page 747).

`VoltaBracket` objects are created by the following engraver(s): `Volta_engraver` (page 507).

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-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.

`range-collapse-threshold` (non-negative, exact integer):

3

If the length of a volta range is greater than or equal to this threshold, print it with a dash. For example, if this is 3, a `\volta 1,2,3` is printed as `'1.-3.'`, but if it is 4, it is printed as `'1.2.3.'`.

`shorten-pair` (pair of numbers):

`ly:volta-bracket::calc-shorten-pair`

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`stencil` (stencil):

`ly:volta-bracket-interface::print`

The symbol to print.

`text` (markup):

`volta-bracket-interface::calc-text`

Text markup. See Section “Formatting text” in *Notation Reference*.

`thickness` (number):

1.6

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve’s outline at its thickest point, not counting the diameter of the virtual “pen” that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`vertical-skylines` (pair of skylines):

`#<unpure-pure-container ly:grob::vertical-skylines-from-stencil`

`ly:grob::pure-simple-vertical-skylines-from-extents >`

Two skylines, one above and one below this grob.

`volta-number-offset` (pair of numbers):

`'(1.0 . -0.5)`

The offset of the volta number relative to the upper left corner of the volta bracket.

`word-space` (dimension, in staff space):

0.6

Space to insert between words in texts.

`Y-extent` (pair of numbers):

`#<unpure-pure-container ly:grob::stencil-height volta-bracket-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): `font-interface` (page 774), `grob-interface` (page 780), `horizontal-bracket-interface` (page 785), `line-interface` (page 794), `side-position-interface` (page 818), `spanner-interface` (page 826), `text-interface` (page 837), `volta-bracket-interface` (page 846), and `volta-interface` (page 847).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

### 3.1.163 VoltaBracketSpanner

An auxiliary grob providing a baseline to align consecutive VoltaBracket (page 745), grobs vertically.

VoltaBracketSpanner objects are created by the following engraver(s): Volta\_engraver (page 507).

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):

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 ly:grob::vertical-skylines-from-element-stencils  
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 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):

```
#<unpure-pure-container ly:side-position-interface::y-aligned-side
ly:side-position-interface::pure-y-aligned-side >
```

The vertical amount that this object is moved relative to its Y-parent.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

This object supports the following interface(s): axis-group-interface (page 753), grob-interface (page 780), outside-staff-interface (page 808), side-position-interface (page 818), spanner-interface (page 826), and volta-interface (page 847).

This object is of class Spanner (characterized by spanner-interface (page 826)).

### 3.1.164 VowelTransition

A vowel transition in lyrics. See also LyricHyphen (page 636).

VowelTransition objects are created by the following engraver(s): Hyphen\_engraver (page 477).

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 ly:grob::vertical-skylines-from-stencil`  
`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.  
 Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

This object supports the following interface(s): `grob-interface` (page 780), `horizontal-line-spanner-interface` (page 786), `line-interface` (page 794), `lyric-interface` (page 797), and `spanner-interface` (page 826).

This object is of class `Spanner` (characterized by `spanner-interface` (page 826)).

## 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`.

`glyph-name` (string)  
 The glyph name within the font.  
 In the context of (span) bar lines or clefs, `glyph-name` represents a processed form of `glyph`, where decisions about line breaking, etc., are already taken.

`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 526), `AccidentalCautionary` (page 527), `AccidentalSuggestion` (page 529), `AmbitusAccidental` (page 533), and `TrillPitchAccidental` (page 730).

### 3.2.2 accidental-participating-head-interface

A grob that should set the current alteration for a pitch in a measure.

This grob interface is used in the following graphical object(s): `NoteHead` (page 658), and `TrillPitchHead` (page 733).

### 3.2.3 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 528).

### 3.2.4 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 529).

### 3.2.5 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 526), `AccidentalCautionary` (page 527), `AccidentalSuggestion` (page 529), `AmbitusAccidental` (page 533), `BalloonText` (page 537), `BassFigure` (page 544), `ChordName` (page 563), `CombineTextScript` (page 572), `GridChordName` (page 612), `HorizontalBracketText` (page 617), `InstrumentName` (page 618), `InstrumentSwitch` (page 619), `KeyCancellation` (page 622), `KeySignature` (page 625), `MeasureSpanner` (page 644), `NoteName` (page 659), `RehearsalMark` (page 673), `TextMark` (page 719), `TextScript` (page 721), and `TrillPitchAccidental` (page 730).

### 3.2.6 align-interface

Order grobs from top to bottom, left to right, right to left or bottom to top. For vertical alignments of staves, the `line-break-system-details` of the left Section “NonMusicalPaperColumn” in *Internals Reference* may be set to tune vertical spacing.

#### User-settable properties:

`align-dir` (direction)  
Which side to align? -1: left side, 0: around center of width, 1: right side.

`axes` (list)  
List of axis numbers. In the case of alignment grobs, this should contain only one number.

`padding` (dimension, in staff space)  
Add this much extra space between objects that are next to each other.

`stacking-dir` (direction)  
Stack objects in which direction?

#### Internal properties:

`elements` (array of grobs)  
An array of grobs; the type is depending on the grob where this is set in.

`minimum-translations-alist` (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 544), and `VerticalAlignment` (page 742).



### 3.2.7 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 531), *AmbitusLine* (page 533), and *AmbitusNoteHead* (page 534).

### 3.2.8 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 a bracket indicating non-arpeggio or optional material, 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 535).

### 3.2.9 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 531), `BassFigureAlignment` (page 544), `BassFigureAlignmentPositioning` (page 545), `BassFigureLine` (page 548), `BreakAlignGroup` (page 554), `BreakAlignment` (page 555), `CenteredBarNumberLineSpanner` (page 562), `DotColumn` (page 588), `DynamicLineSpanner`

(page 595), NonMusicalPaperColumn (page 655), NoteCollision (page 656), NoteColumn (page 657), PaperColumn (page 665), SostenutoPedalLineSpanner (page 692), SustainPedalLineSpanner (page 711), System (page 712), TrillPitchGroup (page 731), UnaCordaPedalLineSpanner (page 740), VerticalAlignment (page 742), VerticalAxisGroup (page 742), and VoltaBracketSpanner (page 747).

### 3.2.10 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 537), and Footnote (page 606).

### 3.2.11 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.

Full-height bar lines are normally squared to meet the outer staff lines, but their ends may be rounded by setting the rounded property. The ends of short and tick bars are always rounded.

### 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 or clefs, *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*).

right-justified (boolean)

Used for BarLines to right-align them. Usually the extent of a BarLine has some positive value to the right. If this property is set to #t, BarLine.stencil is translated to the left by this value. Needs to be set at Score or StaffGroup level. As a result all BarLines of said Score or StaffGroup are right-justified.

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*).

`short-bar-extent` (pair of numbers)

The Y-extent of a short bar line. The default is half the normal bar extent, rounded up to an integer number of staff spaces.

`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:

`allow-span-bar-above` (boolean)

If false, no inter-staff bar line will be created above this item.

`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 538), and `SpanBar` (page 694).

#### 3.2.12 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 542), `CenteredBarNumber` (page 561), and `CenteredBarNumberLineSpanner` (page 562).

#### 3.2.13 bass-figure-alignment-interface

Align a bass figure.

This grob interface is used in the following graphical object(s): `BassFigureAlignment` (page 544).

#### 3.2.14 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 544).

#### 3.2.15 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.

`beam-eps`

Epsilon for beam quant code to check for presence in gap.

`collision-padding`

Padding value to avoid vertical collision with other objects.

`collision-penalty`

Demerit penalty for collision-padding.

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.

ideal-slope-factor

Demerit scaling factor for difference between beam slope and damping slope.

musical-direction-factor

Demerit scaling factor for difference between beam slope and music slope.

region-size

Size of region for checking quant scores.

round-to-zero-slope

Damping slope which is considered zero for purposes of calculating direction penalties.

secondary-beam-demerit

Demerit used in quanting calculations for multiple beams.

stem-collision-factor

Demerit factor used for colliding stems.

stem-length-demerit-factor

Demerit factor used for inappropriate stem lengths.

stem-length-limit-penalty

Penalty for differences in stem lengths on a beam.

### User-settable properties:

accidental-padding (number)

Property used by Beam to avoid accidentals in whole note tremolos.

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)

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

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)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

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.

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.

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 549).

### 3.2.16 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 551).

### 3.2.17 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.

`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.

`bend-arrowhead-height`

The height of the arrow head.

`bend-arrowhead-width`

The width of the arrow head.

`curvature-factor`

Determines the horizontal part of a bent arrow as percentage of the total horizontal extent, usually between 0 and 1.

`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.

`vertical-padding`

Vertical padding between note heads and bends for pre-bend and pre-bend-hold styles.

`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)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values

of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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 552), `NoteColumn` (page 657), `NoteHead` (page 658), and `TabNoteHead` (page 717).

### 3.2.18 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 `#t` 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 629), `PhrasingSlur` (page 670), `RepeatTie` (page 676), `Slur` (page 688), and `Tie` (page 725).

### 3.2.19 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 542), `CodaMark` (page 570), `JumpScript` (page 621), `LyricRepeatCount` (page 637), `MetronomeMark` (page 647), `RehearsalMark` (page 673), `SectionLabel` (page 681), `SegnoMark` (page 683), and `TextMark` (page 719).

### 3.2.20 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)

If `space-alist` is defined for a grob that gets spaced in a staff, an entry for `first-note` must be present. If there is no `next-note` entry, the value of `first-note` is used instead.

Choices for `spacing-style` are:

`extra-space`

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

This grob interface is used in the following graphical object(s): `Ambitus` (page 531), `AmbitusAccidental` (page 533), `BarLine` (page 538), `BreakAlignGroup` (page 554), `BreathingSign` (page 557), `Clef` (page 565), `CueClef` (page 577), `CueEndClef` (page 580), `Custos` (page 583), `Divisio` (page 585), `DoublePercentRepeat` (page 590), `KeyCancellation` (page 622), `KeySignature` (page 625), `LeftEdge` (page 632), `OptionalMaterialBracket` (page 661), `SignumRepetitionis` (page 685), `SpanBar` (page 694), `StaffEllipsis` (page 696), and `TimeSignature` (page 727).

### 3.2.21 break-alignment-interface

The object that performs break alignment.

Three interfaces deal specifically with break alignment:

1. `break-alignment-interface` (this one),
2. `break-alignable-interface` (page 762), and
3. `break-aligned-interface` (page 762).

Each of these interfaces supports grob properties that use *break-align symbols*, which are Scheme symbols 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:

```
\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 555).

### 3.2.22 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 557), and `Divisio` (page 585).

### 3.2.23 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 559).

### 3.2.24 centered-bar-number-interface

A measure-centered bar number.

This grob interface is used in the following graphical object(s): `CenteredBarNumber` (page 561).

### 3.2.25 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 562).

### 3.2.26 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. If not a number, align on the object's reference point.

`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 561), `MeasureCounter` (page 641), and `PercentRepeat` (page 667).

### 3.2.27 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 563), and `FretBoard` (page 608).

### 3.2.28 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 564).

**3.2.29 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 or clefs, *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 565), `CueClef` (page 577), and `CueEndClef` (page 580).

**3.2.30 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 568).

**3.2.31 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 570).



### 3.2.32 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 569).

### 3.2.33 coda-mark-interface

A coda sign.

This grob interface is used in the following graphical object(s): `CodaMark` (page 570).

### 3.2.34 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 575).

### 3.2.35 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 576).

**3.2.36 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 583).

**3.2.37 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.

`padding` (dimension, in staff space)

Add this much extra space between objects that are next to each other.

**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 588).

### 3.2.38 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.

`glyph-name` (string)

The glyph name within the font.

In the context of (span) bar lines or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

`style` (symbol)

This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

#### Internal properties:

`dot-stencil` (stencil)

The stencil for an individual dot, as opposed to a group of several dots.

This grob interface is used in the following graphical object(s): Dots (page 588).

### 3.2.39 duration-line-interface

A line lasting for the duration of a rhythmic event.

If `bound-details.right.end-style` is set to 'arrow, end the duration line with a right-pointing arrow. If set to 'hook, end it with a hook.

The following properties may be set in the details list.

`extra-dot-padding`

Padding to apply if a DotColumn grob is present and the `start-at-dot` sub-property is enabled.

`hook-direction`

The direction of the hook ending the duration line.

`hook-height`

The height of the hook ending the duration line.

`hook-thickness`

The thickness of the hook ending the duration line.

#### User-settable properties:

`details` (alist, with symbols as keys)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's details property. See Section 3.2 [Graphical Object Interfaces],

page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob’s description section.

This grob interface is used in the following graphical object(s): `DurationLine` (page 593).

### 3.2.40 `dynamic-interface`

Any kind of loudness sign.

This grob interface is used in the following graphical object(s): `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), and `Hairpin` (page 614).

### 3.2.41 `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 595).

### 3.2.42 `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 597).

### 3.2.43 `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 599).

### 3.2.44 `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-height* . *right-height*).

`padding` (dimension, in staff space)

Add this much extra space between objects that are next to each other.

`shorten-pair` (pair of numbers)

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`thickness` (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

### Internal properties:

`elements` (array of grobs)

An array of grobs; the type is depending on the grob where this is set in.

This grob interface is used in the following graphical object(s): *BassFigureBracket* (page 546).

#### 3.2.45 episema-interface

An episema line.

This grob interface is used in the following graphical object(s): *Episema* (page 600).

#### 3.2.46 figured-bass-continuation-interface

Simple extender line between bounds.

### Internal properties:

`figures` (array of grobs)

Figured bass objects for continuation line.

This grob interface is used in the following graphical object(s): *BassFigureContinuation* (page 547).

#### 3.2.47 finger-glide-interface

The line between 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)**

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

**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 601).

**3.2.48 finger-interface**

A fingering instruction.

This grob interface is used in the following graphical object(s): `Fingering` (page 603).

**3.2.49 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 605).

**3.2.50 flag-interface**

A flag that gets attached to a stem. The style property is a symbol determining what style of flag glyph is typeset on a Stem grob. Valid options include '()' (for standard flags), 'mensural', 'stacked', 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 or clefs, *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 605).

**3.2.51 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 `serif`, `sans` and `typewriter`.

font-features (list)

OpenType features.

font-name (string)

This property is kept for backwards compatibility only. Use the `fonts` property instead.

**font-series (symbol)**

Select the series of a font. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

**font-shape (symbol)**

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

**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.

**font-stretch (symbol)**

Select a condensed or expanded font, if available in the font family. Possible values are ultra-condensed, extra-condensed, condensed, semi-condensed, normal, semi-expanded, expanded, extra-expanded, and ultra-expanded.

**font-variant (symbol)**

Select the variant of a font. Choices include normal and small-caps.

**fonts (alist, with symbols as keys)**

An alist mapping font families to font names. The standard font families are music, serif, sans and typewriter.

**Internal properties:****font (font metric)**

A cached font metric object.

This grob interface is used in the following graphical object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalSuggestion (page 529), AmbitusAccidental (page 533), AmbitusLine (page 533), AmbitusNoteHead (page 534), Arpeggio (page 535), BalloonText (page 537), BarLine (page 538), BarNumber (page 542), BassFigure (page 544), BendSpanner (page 552), BreathingSign (page 557), CaesuraScript (page 559), CenteredBarNumber (page 561), ChordName (page 563), Clef (page 565), ClefModifier (page 568), CodaMark (page 570), CombineTextScript (page 572), CueClef (page 577), CueEndClef (page 580), Custos (page 583), Divisio (page 585), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DurationLine (page 593), DynamicText (page 597), DynamicTextSpanner (page 599), Episema (page 600), FingerGlideSpanner (page 601), Fingering (page 603), Flag (page 605), Footnote (page 606), FretBoard (page 608), Glissando (page 610), GridChordName (page 612), HorizontalBracketText (page 617), InstrumentName (page 618), InstrumentSwitch (page 619), JumpScript (page 621), KeyCancellation (page 622), KeySignature (page 625), KievanLigature (page 628), LyricHyphen (page 636), LyricRepeatCount (page 637), LyricText (page 640), MeasureCounter (page 641), MeasureSpanner (page 644), MensuralLigature (page 646), MetronomeMark (page 647), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NonMusicalPaperColumn (page 655), NoteHead (page 658), NoteName (page 659), OttavaBracket (page 664), PaperColumn (page 665), Parentheses (page 666), PercentRepeat (page 667), PercentRepeatCounter (page 668), RehearsalMark



(page 673), Rest (page 678), Script (page 679), SectionLabel (page 681), SegnoMark (page 683), SignumRepetitionis (page 685), SostenutoPedal (page 690), SpanBar (page 694), StaffEllipsis (page 696), StanzaNumber (page 702), StringNumber (page 706), StrokeFinger (page 708), SustainPedal (page 710), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TabNoteHead (page 717), TextMark (page 719), TextScript (page 721), TextSpanner (page 723), TimeSignature (page 727), TrillPitchAccidental (page 730), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletNumber (page 737), UnaCordaPedal (page 739), VaticanaLigature (page 741), and VoltaBracket (page 745).

### 3.2.52 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 606).

### 3.2.53 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.
- barre-thickness – Thickness of barre line, in multiples of dot-radius. Only defined for barre-type=straight. Default value 1.

- `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-overhang` – Extension of string lines beyond last fret line, in multiples of fret-distance. Default value 1.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string  $k$  is given by  $\text{thickness} * (1 + \text{string-thickness-factor}) ^ (k-1)$ . Default 0.

- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`size` (number)

The ratio of the size of the object to its default size.

`thickness` (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve’s outline at its thickest point, not counting the diameter of the virtual “pen” that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

This grob interface is used in the following graphical object(s): *FretBoard* (page 608).

### 3.2.54 **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 610).

### 3.2.55 **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 611).

### 3.2.56 **gregorian-ligature-interface**

A gregorian ligature.

#### Internal properties:

`ascendens` (boolean)

Is this neume of ascending type?

`auctum` (boolean)

Is this neume liquescentically augmented?

`cavum` (boolean)

Is this neume outlined?

`context-info` (integer)

Within a ligature, the final glyph or shape of a head may be affected by the left and/or right neighbour head. `context-info` holds for each head such information about the left and right neighbour, encoded as a bit mask.

`deminutum` (boolean)

Is this neume deminished?

`descendens` (boolean)

Is this neume of descendent type?

`inclinatum` (boolean)

Is this neume an inclinatum?

`linea` (boolean)

Attach vertical lines to this neume?

`oriscus` (boolean)

Is this neume an oriscus?

`pes-or-flexa` (boolean)

Shall this neume be joined with the previous head?

`prefix-set` (number)

A bit mask that holds all Gregorian head prefixes, such as `\virga` or `\quilisma`.

`quilisma` (boolean)

Is this neume a quilisma?

`strophia` (boolean)

Is this neume a strophia?

`virga` (boolean)

Is this neume a virga?

This grob interface is used in the following graphical object(s): `NoteHead` (page 658).

### 3.2.57 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 612).

### 3.2.58 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 612).

**3.2.59 grid-point-interface**

A spanning point for grid lines.

This grob interface is used in the following graphical object(s): `GridPoint` (page 613).

**3.2.60 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 `NoteCollision` (page 656), 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 example, '(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. If `whiteout-color` is set, use this color instead of white for the background.

`whiteout-color` (color)

The background color used if property whiteout is set.

`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.

Note that many objects have special positioning considerations, which cause any setting of X-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

`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.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

## 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 526), `AccidentalCautionary` (page 527), `AccidentalPlacement` (page 528), `AccidentalSuggestion` (page 529), `Ambitus` (page 531), `AmbitusAccidental` (page 533), `AmbitusLine` (page 533), `AmbitusNoteHead` (page 534), `Arpeggio` (page 535), `BalloonText` (page 537), `BarLine` (page 538), `BarNumber` (page 542), `BassFigure` (page 544), `BassFigureAlignment` (page 544), `BassFigureAlignmentPositioning` (page 545), `BassFigureBracket` (page 546), `BassFigureContinuation` (page 547), `BassFigureLine` (page 548), `Beam` (page 549), `BendAfter` (page 551), `BendSpanner` (page 552), `BreakAlignGroup` (page 554), `BreakAlignment` (page 555), `BreathingSign` (page 557), `CaesuraScript` (page 559), `CenteredBarNumber` (page 561), `CenteredBarNumberLineSpanner` (page 562), `ChordName` (page 563), `ChordSquare` (page 564), `Clef` (page 565), `ClefModifier` (page 568), `ClusterSpanner` (page 569), `ClusterSpannerBeacon` (page 570), `CodaMark` (page 570), `CombineTextScript` (page 572), `ControlPoint` (page 575), `ControlPolygon` (page 576), `CueClef` (page 577), `CueEndClef` (page 580), `Custos` (page 583), `Divisio` (page 585), `DotColumn` (page 588), `Dots` (page 588), `DoublePercentRepeat` (page 590), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DurationLine` (page 593), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `Episema` (page 600), `FingerGlideSpanner` (page 601), `Fingering` (page 603), `FingeringColumn` (page 605), `Flag` (page 605), `Footnote` (page 606), `FretBoard` (page 608), `Glissando` (page 610), `GraceSpacing` (page 611), `GridChordName` (page 612), `GridLine` (page 612), `GridPoint` (page 613), `Hairpin` (page 614), `HorizontalBracket` (page 615), `HorizontalBracketText` (page 617), `InstrumentName` (page 618), `InstrumentSwitch` (page 619), `JumpScript` (page 621), `KeyCancellation` (page 622), `KeySignature` (page 625), `KievanLigature` (page 628), `LaissezVibrerTie` (page 629), `LaissezVibrerTieColumn` (page 630), `LedgerLineSpanner` (page 631), `LeftEdge` (page 632), `LigatureBracket` (page 634), `LyricExtender` (page 635), `LyricHyphen` (page 636), `LyricRepeatCount` (page 637), `LyricSpace` (page 639), `LyricText` (page 640), `MeasureCounter` (page 641),



MeasureGrouping (page 643), MeasureSpanner (page 644), MelodyItem (page 646), MensuralLigature (page 646), MetronomeMark (page 647), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), NonMusicalPaperColumn (page 655), NoteCollision (page 656), NoteColumn (page 657), NoteHead (page 658), NoteName (page 659), NoteSpacing (page 660), OptionalMaterialBracket (page 661), OttavaBracket (page 664), PaperColumn (page 665), Parentheses (page 666), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), PianoPedalBracket (page 672), RehearsalMark (page 673), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), RestCollision (page 679), Script (page 679), ScriptColumn (page 680), ScriptRow (page 681), SectionLabel (page 681), SegnoMark (page 683), SignumRepetitionis (page 685), Slur (page 688), SostenuatoPedal (page 690), SostenuatoPedalLineSpanner (page 692), SpacingSpanner (page 693), SpanBar (page 694), SpanBarStub (page 695), StaffEllipsis (page 696), StaffGroupier (page 698), StaffHighlight (page 699), StaffSpacing (page 700), StaffSymbol (page 701), StanzaNumber (page 702), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), SustainPedal (page 710), SustainPedalLineSpanner (page 711), System (page 712), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TabNoteHead (page 717), TextMark (page 719), TextScript (page 721), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TimeSignature (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), UnaCordaPedal (page 739), UnaCordaPedalLineSpanner (page 740), VaticanaLigature (page 741), VerticalAlignment (page 742), VerticalAxisGroup (page 742), VoiceFollower (page 744), VoltaBracket (page 745), VoltaBracketSpanner (page 747), and VowelTransition (page 748).

### 3.2.61 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 614).

**3.2.62 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 742).

**3.2.63 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.

`break-overshoot` (pair of numbers)

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

`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)

A pointer to the text grob of 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 615), `OttavaBracket` (page 664), and `VoltaBracket` (page 745).

**3.2.64 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 617).

**3.2.65 horizontal-line-spanner-interface**

This interface is a subset of the `line-spanner-interface` (page 795), 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.

**User-settable properties:**

- `bound-details` (alist, with symbols as keys)  
An alist of properties for determining attachments of spanners to edges.
- `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): *BassFigureContinuation* (page 547), *DurationLine* (page 593), *DynamicTextSpanner* (page 599), *Episema* (page 600), *TextSpanner* (page 723), *TrillSpanner* (page 734), and *VowelTransition* (page 748).

**3.2.66 horizontal-script-interface**

Script placed horizontally left or right from a *NoteHead*. It may need ledger lines.

**User-settable properties:**

- `length-fraction` (number)  
Multiplier for lengths. Used for determining ledger lines and stem lengths.
- `no-ledgers` (boolean)  
If set, don't draw ledger lines on this object.

This grob interface is used in the following graphical object(s): *Script* (page 679).

**3.2.67 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 526), *AccidentalCautionary* (page 527), and *TrillPitchAccidental* (page 730).

### 3.2.68 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`.
  - `barre-thickness` – Thickness of barre line, in multiples of `dot-radius`. Only defined for `barre-type=straight`. Default value 1.
- `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-overhang` – Extension of string lines beyond last fret line, in multiples of `fret-distance`. Default value 1.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string  $k$  is given by  $\text{thickness} * (1 + \text{string-thickness-factor})^{(k-1)}$ . Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`graphical` (boolean)

Display in graphical (vs. text) form.

`harp-pedal-details` (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 721).

### 3.2.69 item-interface

Grobs can be distinguished in their role in the horizontal spacing. Many grobs define constraints on the spacing by their sizes, for example, note heads, clefs, stems, and all other symbols with a fixed shape. These grobs form a subtype called *Item*.

Some items need special treatment for line breaking. For example, a clef is normally only printed at the start of a line (i.e., after a line break). To model this, ‘breakable’ items (clef, key signature, bar lines, etc.) are copied twice. Then we have three versions of each breakable item: one version if there is no line break, one version that is printed before the line break (at the end of a system), and one version that is printed after the line break.

Whether these versions are visible and take up space is determined by the outcome of the *break-visibility* grob property, which is a function taking a direction (-1, 0 or 1) as an argument. It returns a cons of booleans, signifying whether this grob should be transparent and have no extent.

The following variables for *break-visibility* are predefined:

| grob will show:                | before |       |       |
|--------------------------------|--------|-------|-------|
|                                | no     | after |       |
|                                | break  | break | break |
| <i>all-invisible</i>           | no     | no    | no    |
| <i>begin-of-line-visible</i>   | no     | no    | yes   |
| <i>end-of-line-visible</i>     | yes    | no    | no    |
| <i>all-visible</i>             | yes    | yes   | yes   |
| <i>begin-of-line-invisible</i> | yes    | yes   | no    |
| <i>end-of-line-invisible</i>   | no     | yes   | yes   |
| <i>center-invisible</i>        | yes    | no    | yes   |

### User-settable properties:

*break-visibility* (vector)

A vector of 3 booleans, *#(end-of-line unbroken begin-of-line)*. *#t* means visible, *#f* means killed.

*extra-spacing-height* (pair of numbers)

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to *(-inf.0 . +inf.0)*.

*extra-spacing-width* (pair of numbers)

In the horizontal spacing problem, we pad each item by this amount (by adding the ‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to *(+inf.0 . -inf.0)*.

non-musical (boolean)

True if the grob belongs to a NonMusicalPaperColumn.

This grob interface is used in the following graphical object(s): Accidental (page 526), AccidentalCautionary (page 527), AccidentalPlacement (page 528), AccidentalSuggestion (page 529), Ambitus (page 531), AmbitusAccidental (page 533), AmbitusLine (page 533), AmbitusNoteHead (page 534), Arpeggio (page 535), BarLine (page 538), BarNumber (page 542), BassFigure (page 544), BassFigureBracket (page 546), BreakAlignGroup (page 554), BreakAlignment (page 555), BreathingSign (page 557), CaesuraScript (page 559), ChordName (page 563), Clef (page 565), ClefModifier (page 568), ClusterSpannerBeacon (page 570), CodaMark (page 570), CombineTextScript (page 572), CueClef (page 577), CueEndClef (page 580), Custos (page 583), Divisio (page 585), DotColumn (page 588), Dots (page 588), DoublePercentRepeat (page 590), DoublePercentRepeatCounter (page 591), DoubleRepeatSlash (page 592), DynamicText (page 597), Fingering (page 603), FingeringColumn (page 605), Flag (page 605), FretBoard (page 608), GridLine (page 612), GridPoint (page 613), InstrumentSwitch (page 619), JumpScript (page 621), KeyCancellation (page 622), KeySignature (page 625), LaissezVibrerTie (page 629), LaissezVibrerTieColumn (page 630), LeftEdge (page 632), LyricRepeatCount (page 637), LyricText (page 640), MelodyItem (page 646), MetronomeMark (page 647), NonMusicalPaperColumn (page 655), NoteCollision (page 656), NoteColumn (page 657), NoteHead (page 658), NoteName (page 659), NoteSpacing (page 660), OptionalMaterialBracket (page 661), PaperColumn (page 665), RehearsalMark (page 673), RepeatSlash (page 675), RepeatTie (page 676), RepeatTieColumn (page 677), Rest (page 678), RestCollision (page 679), Script (page 679), ScriptColumn (page 680), ScriptRow (page 681), SectionLabel (page 681), SegnoMark (page 683), SignumRepetitionis (page 685), SostenutoPedal (page 690), SpanBar (page 694), SpanBarStub (page 695), StaffEllipsis (page 696), StaffSpacing (page 700), StanzaNumber (page 702), Stem (page 703), StemStub (page 705), StemTremolo (page 705), StringNumber (page 706), StrokeFinger (page 708), SustainPedal (page 710), TabNoteHead (page 717), TextMark (page 719), TextScript (page 721), TimeSignature (page 727), TrillPitchAccidental (page 730), TrillPitchGroup (page 731), TrillPitchHead (page 733), TrillPitchParentheses (page 733), and UnaCordaPedal (page 739).

In addition, this interface is supported conditionally by the following objects depending on their class: BalloonText (page 537), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), and Parentheses (page 666).

### 3.2.70 jump-script-interface

A jump instruction, e.g., *D.S.*.

This grob interface is used in the following graphical object(s): JumpScript (page 621).

### 3.2.71 key-cancellation-interface

A key cancellation.

This grob interface is used in the following graphical object(s): KeyCancellation (page 622).

### 3.2.72 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 622), and `KeySignature` (page 625).

### 3.2.73 kievian-ligature-interface

A kievian 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 628).

### 3.2.74 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 *StaffSymbol* (page 701), 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 631).

### 3.2.75 ledgered-interface

Objects that need ledger lines, typically note heads. See also *ledger-line-spanner-interface* (page 793).

#### 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 534), *NoteHead* (page 658), and *TrillPitchHead* (page 733).

### 3.2.76 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.

### 3.2.77 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 658).

### 3.2.78 ligature-interface

A ligature.

This grob interface is not used in any graphical object.

### 3.2.79 line-interface

Generic line objects. Any object using lines supports this. The property `style` can be `line`, `dashed-line`, `trill`, `dotted-line`, `zigzag` or `none` (a transparent line).

For `dashed-line`, the length of the dashes is tuned with `dash-fraction`. If the latter is set to 0, a dotted line is produced.

### User-settable properties:

`arrow-length` (number)

Arrow length.

`arrow-width` (number)

Arrow width.

`dash-fraction` (number)

Size of the dashes, relative to `dash-period`. Should be between 0.1 and 1.0 (continuous line). If set to 0.0, a dotted line is produced

`dash-period` (number)

The length of one dash together with whitespace. If negative, no line is drawn at all.

`style` (symbol)

This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`thickness` (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`zigzag-length` (dimension, in staff space)

The length of the lines of a zigzag, relative to `zigzag-width`. A value of 1 gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)

The width of one zigzag squiggle. This number is adjusted slightly so that the spanner line can be constructed from a whole number of squiggles.

This grob interface is used in the following graphical object(s): `ChordSquare` (page 564), `DurationLine` (page 593), `DynamicTextSpanner` (page 599), `Episema` (page 600), `Glissando` (page 610), `Hairpin` (page 614), `HorizontalBracket` (page 615), `LigatureBracket` (page 634), `MeasureSpanner` (page 644), `OttavaBracket` (page 664), `PianoPedalBracket` (page 672), `TextSpanner` (page 723), `TrillSpanner` (page 734), `TupletBracket` (page 736), `VoiceFollower` (page 744), `VoltaBracket` (page 745), and `VowelTransition` (page 748).

### 3.2.80 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 `horizontal-line-spanner-interface` (page 786).

`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 552), *FingerGlideSpanner* (page 601), *Glissando* (page 610), and *VoiceFollower* (page 744).

**3.2.81 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 635).

### 3.2.82 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 *Staff.StaffSymbol.thickness*).

This grob interface is used in the following graphical object(s): *LyricHyphen* (page 636), and *LyricSpace* (page 639).

### 3.2.83 lyric-interface

Any object that is related to lyrics.

This grob interface is used in the following graphical object(s): *LyricExtender* (page 635), *LyricHyphen* (page 636), *LyricRepeatCount* (page 637), and *VowelTransition* (page 748).

### 3.2.84 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 637).

**3.2.85 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 639).

**3.2.86 lyric-syllable-interface**

A single piece of lyrics.

This grob interface is used in the following graphical object(s): LyricText (page 640).

**3.2.87 mark-interface**

A rehearsal mark, segno, or coda sign.

This grob interface is used in the following graphical object(s): CodaMark (page 570), RehearsalMark (page 673), SegnoMark (page 683), and TextMark (page 719).

**3.2.88 measure-counter-interface**

A counter for numbering measures.

**User-settable properties:**

`count-from` (integer)

The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`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.

`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 641).

**3.2.89 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 643).

### 3.2.90 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 #f 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)

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

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 644).

### 3.2.91 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 646).

### 3.2.92 mensural-ligature-interface

A mensural ligature.

#### User-settable properties:

`thickness` (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve’s outline at its thickest point, not counting the diameter of the virtual “pen” that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

#### Internal properties:

`add-join` (boolean)

Is this ligature head-joined with the next one by a vertical line?

`delta-position` (number)

The vertical position difference.

`flexa-interval` (integer)

The interval spanned by the two notes of a flexa shape (1 is a second, 7 is an octave).

`head-width` (dimension, in staff space)

The width of this ligature head.

`left-down-stem` (boolean)

request a downward left stem for an initial breve in a ligature.

`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.

`right-down-stem` (boolean)

request a downward right stem for a maxima in a ligature.

`right-up-stem` (boolean)

request an upward right stem for a final longa or maxima in a ligature.

This grob interface is used in the following graphical object(s): `MensuralLigature` (page 646), and `NoteHead` (page 658).

### 3.2.93 metronome-mark-interface

A metronome mark.

This grob interface is used in the following graphical object(s): `MetronomeMark` (page 647).

### 3.2.94 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): `MultiMeasureRest` (page 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), and `MultiMeasureRestText` (page 653).

### 3.2.95 multi-measure-rest-interface

A rest that spans a whole number of measures.

#### User-settable properties:

`bound-padding` (number)

The amount of padding to insert around spanner bounds.

`expand-limit` (integer)

Maximum number of measures expanded in church rests.

`hair-thickness` (number)

Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e., the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`max-symbol-separation` (number)

The maximum distance between symbols making up a church rest.

`measure-count` (integer)

The number of measures for a multi-measure rest.

`minimum-length` (dimension, in staff space)

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`round-up-exceptions` (list)

A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)

Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a  $3/2$  measure.

`spacing-pair` (pair)

A pair of alignment symbols which set an object's spacing relative to its left and right `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 default staff-line thickness (i.e., the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`usable-duration-logs` (list)

List of duration-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 649), and `PercentRepeat` (page 667).

#### 3.2.96 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 650).

#### 3.2.97 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 665).

### 3.2.98 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 555), in a `NonMusicalPaperColumn` (page 655).

This grob interface is used in the following graphical object(s): `NonMusicalPaperColumn` (page 655).

### 3.2.99 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 `note-column-interface` (page 804): 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 656).

**3.2.100 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 down-stem voice note; if there are no down-stem voices, the width of the first up-stem voice note is taken instead. 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 657).

### 3.2.101 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 or clefs, *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 534), `NoteHead` (page 658), `TabNoteHead` (page 717), and `TrillPitchHead` (page 733).

**3.2.102 note-name-interface**

Note names.

This grob interface is used in the following graphical object(s): `NoteName` (page 659).

**3.2.103 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 660).

**3.2.104 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 706).

### 3.2.105 optional-material-bracket-interface

An in-staff bracket around optional material.

#### 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.

`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 a bracket indicating non-arpeggio or optional material, the length of the horizontal 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`).

This grob interface is used in the following graphical object(s): `OptionalMaterialBracket` (page 661).

### 3.2.106 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.

`edge-height` (pair)

A pair of numbers specifying the heights of the vertical edges: (*left-height* . *right-height*).

`minimum-length` (dimension, in staff space)

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a Tie, this sets the minimum distance between noteheads.

`shorten-pair` (pair of numbers)

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

This grob interface is used in the following graphical object(s): `OttavaBracket` (page 664).

### 3.2.107 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 548), `System` (page 712), and `VerticalAxisGroup` (page 742).

**3.2.108 outside-staff-interface**

A grob that could be placed outside staff.

**User-settable properties:**

`outside-staff-horizontal-padding` (number)

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)

The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-priority` (number)

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

This grob interface is used in the following graphical object(s): `AccidentalSuggestion` (page 529), `BarNumber` (page 542), `BassFigureAlignmentPositioning` (page 545), `BendSpanner` (page 552), `BreathingSign` (page 557), `CaesuraScript` (page 559), `CenteredBarNumberLineSpanner` (page 562), `ChordName` (page 563), `ClefModifier` (page 568), `CodaMark` (page 570), `CombineTextScript` (page 572), `Divisio` (page 585), `DoublePercentRepeatCounter` (page 591), `DoubleRepeatSlash` (page 592), `DynamicLineSpanner` (page 595), `DynamicText` (page 597), `Fingering` (page 603), `FretBoard` (page 608), `Hairpin` (page 614), `HorizontalBracket` (page 615), `HorizontalBracketText` (page 617), `InstrumentSwitch` (page 619), `JumpScript` (page 621), `LigatureBracket` (page 634), `MeasureCounter` (page 641), `MeasureGrouping` (page 643), `MeasureSpanner` (page 644), `MetronomeMark` (page 647), `MultiMeasureRest` (page 649), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), `MultiMeasureRestText` (page 653), `OttavaBracket` (page 664), `PercentRepeatCounter` (page 668), `PhrasingSlur` (page 670), `RehearsalMark` (page 673),

Script (page 679), SectionLabel (page 681), SegnoMark (page 683), Slur (page 688), SostenuatoPedalLineSpanner (page 692), StringNumber (page 706), StrokeFinger (page 708), SustainPedalLineSpanner (page 711), TextMark (page 719), TextScript (page 721), TextSpanner (page 723), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), UnaCordaPedalLineSpanner (page 740), and VoltaBracketSpanner (page 747).

### 3.2.109 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 790).

#### 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.

X-alignment-extent (pair of numbers)

If a grob wants to align itself on a PaperColumn grob that doesn't contain note heads, use this horizontal extent as a placeholder.

#### 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 655), and PaperColumn (page 665).

### 3.2.110 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 666), and *TrillPitchParentheses* (page 733).

**3.2.111 passage-delimiter-interface**

Items with this interface allow overriding *break-visibility-passage-start*, *break-visibility-passage-default*, and *break-visibility-passage-end* particularly in lieu of overriding *break-visibility*. The precise meaning of ‘passage’ depends on the type of item.

The internal *passage-direction* property determines which *break-visibility* property is used in a given instance.

See Section “item-interface” in *Internals Reference*.

**User-settable properties:**

*break-visibility-passage-default* (vector)

The value to use for *break-visibility* when the item does not specifically mark the start or end of a passage. (It might be both or neither, depending on the type of item.)

*break-visibility-passage-end* (vector)

The value to use for *break-visibility* when the item marks the end of a passage.

*break-visibility-passage-start* (vector)

The value to use for *break-visibility* when the item marks the start of a passage.

**Internal properties:**

*passage-direction* (direction)

The placement of a *passage-delimiter-interface* item with respect to its passage: *START* at the start, *END* at the end, or *CENTER* otherwise.

This grob interface is used in the following graphical object(s): *OptionalMaterialBracket* (page 661), and *StaffEllipsis* (page 696).

**3.2.112 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 590), *DoubleRepeatSlash* (page 592), *PercentRepeat* (page 667), and *RepeatSlash* (page 675).

### 3.2.113 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.

`edge-height` (pair)

A pair of numbers specifying the heights of the vertical edges: (*left-height* . *right-height*).

`shorten-pair` (pair of numbers)

The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

#### Internal properties:

`pedal-text` (graphical (layout) object)

A pointer to the text grob of a mixed-style piano pedal.

This grob interface is used in the following graphical object(s): *PianoPedalBracket* (page 672).

### 3.2.114 piano-pedal-interface

A piano pedal sign.

This grob interface is used in the following graphical object(s): *PianoPedalBracket* (page 672), *SostenutoPedalLineSpanner* (page 692), *SustainPedal* (page 710), *SustainPedalLineSpanner* (page 711), and *UnaCordaPedalLineSpanner* (page 740).

### 3.2.115 piano-pedal-script-interface

A piano pedal sign, fixed size.

This grob interface is used in the following graphical object(s): *SostenutoPedal* (page 690), *SustainPedal* (page 710), and *UnaCordaPedal* (page 739).

### 3.2.116 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 733), and TrillPitchParentheses (page 733).

**3.2.117 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 538), Clef (page 565), CueClef (page 577), CueEndClef (page 580), KeyCancellation (page 622), KeySignature (page 625), SignumRepetitionis (page 685), SpanBarStub (page 695), and TimeSignature (page 727).

**3.2.118 rehearsal-mark-interface**

A rehearsal mark.

This grob interface is used in the following graphical object(s): RehearsalMark (page 673).

**3.2.119 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 679).

**3.2.120 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 649), and `Rest` (page 678).

**3.2.121 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 544), `ChordName` (page 563), `ClusterSpannerBeacon` (page 570), `DoubleRepeatSlash` (page 592), `FretBoard` (page 608), `LyricText` (page 640), `NoteHead` (page 658), `RepeatSlash` (page 675), `Rest` (page 678), and `TabNoteHead` (page 717).

**3.2.122 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): `NoteHead` (page 658), `Rest` (page 678), and `TabNoteHead` (page 717).

**3.2.123 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 680), and `ScriptRow` (page 681).

**3.2.124 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.

`staff-position` (number)  
Vertical position, measured in half staff spaces, counted from the middle line.  
For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

`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.

`grob-defaults` (list)  
A list with `type` and an alist of properties, derived from a list like `default-script-alist`, p.e.  
(*accent*

```
. ((avoid-slur . around)
 (padding . 0.20)
 (script-stencil . (feta . ("sforzato" . "sforzato"))))
(side-relative-direction . ,DOWN)))
```

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 529), CaesuraScript (page 559), DynamicText (page 597), MultiMeasureRestScript (page 652), and Script (page 679).

### 3.2.125 section-label-interface

A section label, e.g., “Coda”.

This grob interface is used in the following graphical object(s): SectionLabel (page 681).

### 3.2.126 segno-mark-interface

A segno.

This grob interface is used in the following graphical object(s): SegnoMark (page 683).

### 3.2.127 self-alignment-interface

Position this object on itself and/or on its parent. To this end, the following functions are provided:

ly:self-alignment-interface::x-aligned-on-self

ly:self-alignment-interface::y-aligned-on-self

Align self on reference point, using self-alignment-X and self-alignment-Y, respectively.

ly:self-alignment-interface::aligned-on-x-parent

ly:self-alignment-interface::aligned-on-y-parent

Align reference point of self with the reference point of parent. The position of the own reference point is adjusted with self-alignment-X and self-alignment-Y, the position of the parent’s reference point with parent-alignment-X and parent-alignment-Y, respectively.

Function ly:self-alignment-interface::aligned-on-x-parent listens to the property X-alignment-extent of the PaperColumn grob, using it as a fallback width for parent alignment in case the PaperColumn grob does not contain note heads.

ly:self-alignment-interface::centered-on-x-parent

ly:self-alignment-interface::centered-on-y-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 not a number, align on the parent's reference point. 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. If not a number, align on the object's reference point.

`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 529), `BarNumber` (page 542), `CaesuraScript` (page 559), `ClefModifier` (page 568), `CodaMark` (page 570), `CombineTextScript` (page 572), `DoublePercentRepeatCounter` (page 591), `DynamicText` (page 597), `Fingering` (page 603), `GridLine` (page 612), `Hairpin` (page 614), `HorizontalBracketText` (page 617), `InstrumentName` (page 618), `InstrumentSwitch` (page 619), `JumpScript` (page 621), `LyricRepeatCount` (page 637), `LyricText` (page 640), `MeasureCounter` (page 641), `MeasureSpanner` (page 644), `MetronomeMark` (page 647), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestScript` (page 652), `MultiMeasureRestText` (page 653), `NoteName` (page 659), `PercentRepeatCounter` (page 668), `RehearsalMark` (page 673), `Script` (page 679), `SectionLabel` (page 681), `SegnoMark` (page 683), `SostenutoPedal` (page 690), `StemTremolo` (page 705), `StringNumber` (page 706), `StrokeFinger` (page 708), `SustainPedal` (page 710), `TextMark` (page 719), `TextScript` (page 721), and `UnaCordaPedal` (page 739).

### 3.2.128 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 a desired tie configuration that overrides the default. *position* is the offset from the center of the staff in half staff-space units, 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.

There is a distinction between exact and inexact values for *position*: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

#### 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 630), and `RepeatTieColumn` (page 677).

### 3.2.129 semi-tie-interface

A tie that 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)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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 629), and `RepeatTie` (page 676).

**3.2.130 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 655), NoteColumn (page 657), and PaperColumn (page 665).

**3.2.131 side-position-interface**

Position a victim object (this one) next to other objects (the support). The property direction signifies where to put the victim object relative to the support (left or right, up or down?)

The routine also takes the size of the staff into account if staff-padding is set. If undefined, the staff symbol is ignored.

**User-settable properties:**

add-stem-support (boolean)

If set, the Stem object is included in this script's support.

direction (direction)

If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

horizon-padding (number)

The amount to pad the axis along which a Skyline is built for the side-position-interface.

minimum-space (dimension, in staff space)

Minimum distance that the victim should move (after padding).

minimum-X-space (dimension, in staff space)

Minimum distance that the victim should move horizontally (after padding), overriding the minimum-space property value.

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.

X-padding (dimension, in staff space)

Add this much extra space between objects that are next to each other horizontally, overriding the padding property value.

### Internal properties:

quantize-position (boolean)

If set, a vertical alignment is aligned to be within staff spaces.

side-support-elements (array of grobs)

The side support, an array of grobs.

This grob interface is used in the following graphical object(s): *AccidentalSuggestion* (page 529), *Arpeggio* (page 535), *BarNumber* (page 542), *BassFigureAlignmentPositioning* (page 545), *CaesuraScript* (page 559), *CenteredBarNumberLineSpanner* (page 562), *ClefModifier* (page 568), *CodaMark* (page 570), *CombineTextScript* (page 572), *DoublePercentRepeatCounter* (page 591), *DynamicLineSpanner* (page 595), *Episema* (page 600), *Fingering* (page 603), *HorizontalBracket* (page 615), *HorizontalBracketText* (page 617), *InstrumentName* (page 618), *InstrumentSwitch* (page 619), *JumpScript* (page 621), *MeasureCounter* (page 641), *MeasureGrouping* (page 643), *MeasureSpanner* (page 644), *MetronomeMark* (page 647), *MultiMeasureRestNumber* (page 650), *MultiMeasureRestScript* (page 652), *MultiMeasureRestText* (page 653), *OttavaBracket* (page 664), *PercentRepeatCounter* (page 668), *RehearsalMark* (page 673), *Script* (page 679), *SectionLabel* (page 681), *SegnoMark* (page 683), *SostenutoPedalLineSpanner* (page 692), *StanzaNumber* (page 702), *StringNumber* (page 706), *StrokeFinger* (page 708), *SustainPedalLineSpanner* (page 711), *SystemStartBar* (page 713), *SystemStartBrace* (page 714), *SystemStartBracket* (page 715), *SystemStartSquare* (page 716), *TextMark* (page 719), *TextScript* (page 721), *TextSpanner* (page 723), *TrillPitchAccidental* (page 730), *TrillPitchGroup* (page 731), *TrillSpanner* (page 734), *UnaCordaPedalLineSpanner* (page 740), *VoltaBracket* (page 745), and *VoltaBracketSpanner* (page 747).

### 3.2.132 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 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 or clefs, *glyph-name* represents a processed form of glyph, where decisions about line breaking, etc., are already taken.

hair-thickness (number)

Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e., the visual output is *not* influenced by changes to *Staff.StaffSymbol.thickness*).

kern (dimension, in staff space)

The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e., the visual output is *not* influenced by changes to *Staff.StaffSymbol.thickness*).

rounded (boolean)

Decide whether lines should be drawn rounded or not.

segno-kern (number)

The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e., the visual output is *not* influenced by changes to *Staff.StaffSymbol.thickness*).

short-bar-extent (pair of numbers)

The Y-extent of a short bar line. The default is half the normal bar extent, rounded up to an integer number of staff spaces.

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): *SignumRepetitionis* (page 685).

### 3.2.133 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.

absolute-closeness-measure

Factor to calculate demerit for variance between a note head and slur.

**accidental-collision**

Factor to calculate demerit for Accidental objects that the slur encompasses. This property value replaces the value of `extra-object-collision-penalty`.

**close-to-edge-length**

Threshold to decide whether an object to avoid is horizontally close to the slur's edge. If it is, it doesn't influence the slur's height.

**edge-attraction-factor**

Factor used to calculate the demerit for distances between slur endpoints and their corresponding base attachments.

**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.

**encompass-object-range-overshoot**

Widen the range of `encompass-object` positions by this amount for computing the slur.

**extra-encompass-collision-distance**

This detail is currently unused.

**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-object-collision-penalty**

Factor to calculate demerit for extra objects that the slur encompasses, including accidentals, fingerings, and tuplet numbers.

**free-head-distance**

The amount of vertical free space that must exist between a slur and note heads.

**free-slur-distance**

The amount of vertical free space that must exist between adjacent slurs. This subproperty only works for `PhrasingSlur`.

**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.

**head-encompass-penalty**

Demerit to apply when note heads collide with a slur.

**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.

**max-slope**

The maximum slope allowed for this slur.

**max-slope-factor**

Factor that calculates demerit based on the max slope. Notice that there exists a homonymous property for tuplet brackets.

**non-horizontal-penalty**

Demerit for slurs with attachment points that are not horizontally aligned.

`region-size`

Size of region (in staff spaces) for determining potential endpoints in the Y-direction.

`same-slope-penalty`

Demerit for slurs with attachment points that are horizontally aligned.

`slur-tie-extrema-min-distance`

If a slur starts or ends very near to or at the same position as a tie, check this threshold whether slur and tie are too close.

`slur-tie-extrema-min-distance-penalty`

Demerit to apply if `slur-tie-extrema-min-distance` gets triggered.

`steeper-slope-factor`

Factor used to calculate demerit only if this slur is not broken.

`stem-encompass-penalty`

Demerit to apply when stems collide with a slur.

## 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)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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 670), and `Slur` (page 688).

### 3.2.134 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` (positive moment with no grace part)

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 655), and PaperColumn (page 665).

### 3.2.135 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 660), and StaffSpacing (page 700).

### 3.2.136 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 611), and SpacingSpanner (page 693).

### 3.2.137 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 proportional 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 693).

### 3.2.138 `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 or clefs, *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 694).

### 3.2.139 `span-bar-stub-interface`

A stand-in for a bar line that is used when engraving span bars. This is an internal interface even though some of its properties are documented as user properties.

#### User-settable properties:

`allow-span-bar` (boolean)

If false, no inter-staff bar line will be created below this bar line.

#### Internal properties:

`allow-span-bar-above` (boolean)

If false, no inter-staff bar line will be created above this item.

`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): `SpanBarStub` (page 695).

### 3.2.140 `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 544), BassFigureAlignmentPositioning (page 545), BassFigureContinuation (page 547), BassFigureLine (page 548), Beam (page 549), BendAfter (page 551), BendSpanner (page 552), CenteredBarNumber (page 561), CenteredBarNumberLineSpanner (page 562), ChordSquare (page 564), ClusterSpanner (page 569), DurationLine (page 593), DynamicLineSpanner (page 595), DynamicTextSpanner (page 599), Episema (page 600), FingerGlideSpanner (page 601), Glissando (page 610), GraceSpacing (page 611), GridChordName (page 612), Hairpin (page 614), HorizontalBracket (page 615), HorizontalBracketText (page 617), InstrumentName (page 618), KievanLigature (page 628), LedgerLineSpanner (page 631), LigatureBracket (page 634), LyricExtender (page 635), LyricHyphen (page 636), LyricSpace (page 639), MeasureCounter (page 641), MeasureGrouping (page 643), MeasureSpanner (page 644), MensuralLigature (page 646), MultiMeasureRest (page 649), MultiMeasureRestNumber (page 650), MultiMeasureRestScript (page 652), MultiMeasureRestText (page 653), OttavaBracket (page 664), PercentRepeat (page 667), PercentRepeatCounter (page 668), PhrasingSlur (page 670), PianoPedalBracket (page 672), Slur (page 688), SostenuatoPedalLineSpanner (page 692), SpacingSpanner (page 693), StaffGrouper (page 698), StaffHighlight (page 699), StaffSymbol (page 701), SustainPedalLineSpanner (page 711), System (page 712), SystemStartBar (page 713), SystemStartBrace (page 714), SystemStartBracket (page 715), SystemStartSquare (page 716), TextSpanner (page 723), Tie (page 725), TieColumn (page 727), TrillSpanner (page 734), TupletBracket (page 736), TupletNumber (page 737), UnaCordaPedalLineSpanner (page 740), VaticanaLigature (page 741), VerticalAlignment (page 742), VerticalAxisGroup (page 742), VoiceFollower (page 744), VoltaBracket (page 745), VoltaBracketSpanner (page 747), and VowelTransition (page 748).

In addition, this interface is supported conditionally by the following objects depending on their class: BalloonText (page 537), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), and Parentheses (page 666).

### 3.2.141 staff-ellipsis-interface

A visual marker (usually three consecutive dots) to indicate that typesetting of music is skipped.

This grob interface is used in the following graphical object(s): StaffEllipsis (page 696).

### 3.2.142 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 698).

### 3.2.143 `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 699).

### 3.2.144 `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 700).

### 3.2.145 `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 701).

**3.2.146 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.

For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

This grob interface is used in the following graphical object(s): `AmbitusNoteHead` (page 534), `Arpeggio` (page 535), `Beam` (page 549), `Clef` (page 565), `CueClef` (page 577), `CueEndClef` (page 580), `Custos` (page 583), `Dots` (page 588), `KeyCancellation` (page 622), `KeySignature` (page 625), `MultiMeasureRest` (page 649), `NoteHead` (page 658), `Rest` (page 678), `TabNoteHead` (page 717), and `TrillPitchHead` (page 733).

### 3.2.147 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 702).

### 3.2.148 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-extreme-minimum-free-lengths`

List of extreme minimum free stem lengths (chord to beams) given beam multiplicity.

`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.

`lengths`

Default stem lengths. The list gives a length for each flag count. If a list entry is a pair, it gives the stem length for the specific up and down stem, respectively.

`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)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`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.

This grob interface is used in the following graphical object(s): Stem (page 703).

**3.2.149 stem-tremolo-interface**

A beam slashing a stem to indicate a tremolo. The property shape can be beam-like or rectangle.

**User-settable properties:**

- beam-thickness (dimension, in staff space)  
Beam thickness, measured in staff-space units.
- beam-width (dimension, in staff space)  
Width of the tremolo sign.
- direction (direction)  
If side-axis is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.
- flag-count (number)  
The number of tremolo beams.
- length-fraction (number)  
Multiplier for lengths. Used for determining ledger lines and stem lengths.
- shape (symbol)  
This setting determines what shape a grob has. Valid choices depend on the stencil callback reading this property.
- slope (number)  
The slope of this object.

`style` (symbol)

This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

### 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 705).

### 3.2.150 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 537), ControlPoint (page 575), ControlPolygon (page 576), Footnote (page 606), and Parentheses (page 666).

### 3.2.151 string-number-interface

A string number instruction.

This grob interface is used in the following graphical object(s): StringNumber (page 706).

### 3.2.152 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 708).

### 3.2.153 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 826).

### 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-stencil` (stencil)  
The stencil of a system's in-notes.
- `in-note-system-padding` (number)  
Padding between in-note and its associated system.
- `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 712).

**3.2.154 system-start-delimiter-interface**

The brace, bracket or bar in front of the system. The following values for `style` are recognized:

**bracket**

A thick bracket, normally used to group similar instruments in a score. Default for `StaffGroup`. `SystemStartBracket` uses this style.

**brace**

A 'piano style' brace normally used for an instrument that uses two staves. The default style for `GrandStaff`. `SystemStartBrace` uses this style.

**bar-line**

A simple line between the staves in a score. Default for staves enclosed in `<<` and `>>`. `SystemStartBar` uses this style.

**line-bracket**

A simple square, normally used for subgrouping instruments in a score. `SystemStartSquare` uses this style.

See also `input/regression/system-start-nesting.ly`.

**User-settable properties:**

- `collapse-height` (dimension, in staff space)  
Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`style` (symbol)

This setting determines in what style a grob is typeset. Valid choices depend on the stencil callback reading this property.

`thickness` (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e., the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

This grob interface is used in the following graphical object(s): `SystemStartBar` (page 713), `SystemStartBrace` (page 714), `SystemStartBracket` (page 715), and `SystemStartSquare` (page 716).

### 3.2.155 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. If not a number, align on the object's reference point.

`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 618).

### 3.2.156 tab-note-head-interface

A note head in tablature.

The following properties may be set in the `details` list.

`cautionary-properties`

An alist to format cautionaries (usually parentheses enclosing the number) with the following elements.

`angularity`

How much the parentheses should become angular.

`half-thickness`

The maximum thickness of a parenthesis.

`padding`

The padding between the parentheses and the enclosed number.

`procedure`

A function to handle cautionaries, taking the other four elements of the `cautionary-properties` alist as arguments.

`width`

The maximum horizontal extent of a parenthesis.

`harmonic-properties`

An alist to format harmonics (usually parentheses enclosing the number) with the following elements.

`angularity`

How much the parentheses should become angular.

`half-thickness`

The maximum thickness of a parenthesis.

`padding`

The padding between the parentheses and the enclosed number.

`procedure`

A function to handle harmonics, taking the other four elements of the `harmonic-properties` alist as arguments.

`width`

The maximum horizontal extent of a parenthesis.

`head-offset`

Move all tablature numbers horizontally. The value is given as a multiple of a single-digit number width.

`tied-properties`

An alist with the following elements.

`tied`

Keep track whether a Tie ends at this note.

`repeat-tied`

Does this note hold a RepeatTie? 'repeat-tied is also set if the note is part of a chord with set RepeatTie.

`note-head-visible`

If set to `#t`, show a number for a note with `\repeatTie` or a note ending a Tie.

`parenthesize`

If set to `#t`, parenthesize the number for a tied note after a line break.

## User-settable properties:

`details` (alist, with symbols as keys)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`parenthesized` (boolean)

Parenthesize this grob.

## Internal properties:

`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 717).

### 3.2.157 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 flags to be displayed within markups (via `\note-by-number`). Available are 'modern-straight-flag, 'old-straight-flag, 'flat-flag, 'mensural, 'stacked, 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 537), `BarNumber` (page 542), `BassFigure` (page 544), `BendSpanner` (page 552), `BreathingSign` (page 557), `CenteredBarNumber` (page 561), `ChordName` (page 563), `ClefModifier` (page 568), `CodaMark` (page 570), `CombineTextScript` (page 572), `ControlPoint` (page 575), `ControlPolygon` (page 576), `Divisio` (page 585), `DoublePercentRepeatCounter` (page 591), `DynamicText` (page 597), `DynamicTextSpanner` (page 599), `Fingering` (page 603), `Footnote` (page 606), `GridChordName` (page 612), `HorizontalBracketText` (page 617), `InstrumentName` (page 618), `InstrumentSwitch` (page 619), `JumpScript` (page 621), `LyricRepeatCount` (page 637), `LyricText` (page 640), `MeasureCounter` (page 641), `MeasureSpanner` (page 644), `MetronomeMark` (page 647), `MultiMeasureRestNumber` (page 650), `MultiMeasureRestText` (page 653), `NoteName` (page 659), `OttavaBracket` (page 664), `PercentRepeatCounter` (page 668), `RehearsalMark` (page 673), `SectionLabel` (page 681), `SegnoMark` (page 683), `SostenutoPedal` (page 690), `StaffEllipsis` (page 696), `StanzaNumber` (page 702), `StringNumber` (page 706), `StrokeFinger` (page 708), `SustainPedal` (page 710), `TabNoteHead` (page 717), `TextMark` (page 719), `TextScript` (page 721), `TupletNumber` (page 737), `UnaCordaPedal` (page 739), and `VoltaBracket` (page 745).

### 3.2.158 text-mark-interface

A textual mark.

This grob interface is used in the following graphical object(s): `TextMark` (page 719).

### 3.2.159 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 552), `CombineTextScript` (page 572), `Fingering` (page 603), `StringNumber` (page 706), `StrokeFinger` (page 708), and `TextScript` (page 721).

**3.2.160 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 a desired tie configuration that overrides the default. *position* is the offset from the center of the staff in half staff-space units, 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.

There is a distinction between exact and inexact values for *position*: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

**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 727).

**3.2.161 tie-interface**

A tie – a horizontal curve connecting two noteheads.

The following properties may be set in the `details` list.

`between-length-limit`

This detail is currently unused.

**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.

**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.

**height-limit**

The maximum height allowed for this tie.

**horizontal-distance-penalty-factor**

Demerit factor for ties in the set being considered which are horizontally distant from the note heads.

**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.

**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.

**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.

**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.

**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.

**ratio**

Parameter for tie shape. The higher this number, the quicker the slur attains its height limit.

**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.

**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.

**skyline-padding**

Padding of the skylines around note heads in chords.



**staff-line-collision-penalty**

Demerit factor for ties whose tips or center come close to staff lines.

**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.

**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.

**vertical-distance-penalty-factor**

Demerit factor for ties in the set being considered which are vertically distant from the note heads.

**wrong-direction-offset-penalty**

Demerit for ties that are offset in the wrong direction.

**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)**

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's *details* property. See Section 3.2 [Graphical Object Interfaces], page 749, for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

**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.

For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

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 629), *RepeatTie* (page 676), and *Tie* (page 725).

### 3.2.162 time-signature-interface

A time signature, in different styles. The following values of style are recognized:

C 4/4 and 2/2 are typeset as C and struck C, respectively. All other time signatures are written as two numbers. 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 as two numbers.

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 as two numbers.

single-number

All time signatures are typeset as a single number, e.g., 3/2 is written as 3.

numbered

All time signatures are typeset as two numbers.

When style allows printing denominators, denominator-style can alter them. The following values are recognized:

none

Do not print denominators.

**note**

Print denominators as a note and dots when exact representation is possible; otherwise print them as numbers.

**number**

Print denominators as numbers. This is the default.

When the denominator is a note, `note-flag-style` and `note-head-style` control its appearance, and `note-dots-direction` and `note-staff-position` control its placement.

**User-settable properties:**

`denominator-style` (symbol)

The style of denominators in a time signature.

`fraction` (pair of numbers)

Numerator and denominator of a time signature object.

`nested-fraction-mixed` (boolean)

Whether a fractional term of a time signature is printed as a mixed number (e.g., ‘2 1/2’) or as a common fraction (e.g., ‘5/2’).

`nested-fraction-orientation` (symbol)

A symbol describing the orientation of a fractional part of a time signature.

`nested-fraction-relative-font-size` (number)

The font size of a numeral in a fractional term of a time signature, relative to size of the whole numbers.

`note-dots-direction` (direction)

Whether the augmentation dots are shifted up or down (or not shifted) relative to the note head in a number-over-note time signature.

`note-flag-style` (symbol)

The style of the flags in a number-over-note time signature. See `flag-style`.

`note-head-style` (symbol)

The style of the note head in a number-over-note time signature. See Section “Note head styles” in *Notation Reference*.

`note-staff-position` (number)

The position of the note in a number-over-note time signature. See `staff-position`.

`senza-misura-stencil` (stencil)

The symbol to print when `TimeSignature.fraction` is not set. Overriding `TimeSignature.stencil` circumvents this.

`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 727).

**3.2.163 trill-pitch-accidental-interface**

An accidental for trill pitch.

This grob interface is used in the following graphical object(s): `TrillPitchAccidental` (page 730).

**3.2.164 trill-spanner-interface**

A trill spanner.

This grob interface is used in the following graphical object(s): `TrillSpanner` (page 734).

### 3.2.165 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 `#f` 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)

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

`connect-to-neighbor` (pair)

Pair of booleans, indicating whether this grob looks as a continued break.

`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`.

`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.

max-slope-factor (non-negative number)

Factor for calculating the maximum tuplet bracket slope. Notice that there exists a homonymous property for slurs.

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.

span-all-note-heads (boolean)

If true, tuplet brackets are printed spanning horizontally from the first to the last note head instead of covering only the stems.

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.

visible-over-note-heads (boolean)

This prints a tuplet bracket when the bracket is set to be over the note heads. This option can be combined with the default tuplet bracket visibility style and with `#'if-no-beam`.

X-positions (pair of numbers)

Pair of X staff coordinates of a spanner in the form (*left* . *right*), where both *left* and *right* are in staff-space units of the current staff.

## Internal properties:

beam (graphical (layout) object)

A pointer to the beam, if applicable.

note-columns (array of grobs)

An array of *NoteColumn* grobs.

potential-beam (graphical (layout) object)

For tuplet brackets, a grob to use as parallel beam unless the tuplet is broken.

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 634), and `TupletBracket` (page 736).

### 3.2.166 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 737).

### 3.2.167 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 549), `DurationLine` (page 593), and `Glissando` (page 610).

### 3.2.168 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 or clefs, *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).

`head-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 658), and *VaticanaLigature* (page 741).

### 3.2.169 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.

`musical-length` (non-negative moment with no grace part)

Musical length.

`range-collapse-threshold` (non-negative, exact integer)

If the length of a volta range is greater than or equal to this threshold, print it with a dash. For example, if this is 3, a `\volta 1,2,3` is printed as '1.-3.', but if it is 4, it is printed as '1.2.3.'.

`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*).

*volta-number-offset* (pair of numbers)

The offset of the volta number relative to the upper left corner of the volta bracket.

### Internal properties:

*bars-left* (array of grobs)

An array of bar line pointers for the left side of a volta bracket.

*bars-right* (array of grobs)

An array of bar line pointers for the right side of a volta bracket.

*volta-numbers* (number list)

List of volta numbers.

This grob interface is used in the following graphical object(s): *VoltaBracket* (page 745).

### 3.2.170 volta-interface

A volta repeat.

This grob interface is used in the following graphical object(s): *VoltaBracket* (page 745), and *VoltaBracketSpanner* (page 747).

## 3.3 User backend properties

*accidental-padding* (number)

Property used by Beam to avoid accidentals in whole note tremolos.

*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 `#f` prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)

Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)

This is a vector of 3 lists:  `#(end-of-line unbroken start-of-line)`. Each list contains *break-align symbols* that specify an order of breakable items (see Section “break-alignment-interface” in *Internals Reference*).

For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
 #(make-vector 3 '(left-edge
 cue-end-clef
 ambitus
 breathing-sign
```

```

time-signature
clef
cue-clef
staff-bar
key-cancellation
key-signature
custos))

```

`break-align-symbol` (symbol)

This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in *Internals Reference*.

`break-align-symbols` (list)

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to break-visibility, we will align to the next grob (and so on). Choices are listed in Section “break-alignment-interface” in *Internals Reference*.

`break-overshoot` (pair of numbers)

A pair of numbers specifying how much a broken spanner sticks out of its bounds horizontally on the broken side(s). For broken beams and broken tuplet brackets, the bounds are given by the prefatory matter on the left and/or the rightmost column on the right. For broken horizontal brackets, the bounds are the leftmost and/or rightmost column; for broken measure spanners, the left and/or right edge of the staff.

`break-visibility` (vector)

A vector of 3 booleans,  `#(end-of-line unbroken begin-of-line)`. `#t` means visible, `#f` means killed.

`break-visibility-passage-default` (vector)

The value to use for break-visibility when the item does not specifically mark the start or end of a passage. (It might be both or neither, depending on the type of item.)

`break-visibility-passage-end` (vector)

The value to use for break-visibility when the item marks the end of a passage.

`break-visibility-passage-start` (vector)

The value to use for break-visibility when the item marks the start of a passage.

`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

`clip-edges` (boolean)

Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)

Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-interfaces` (list)

A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)

Does automatic beam collision apply only to the voice in which the beam was created?

`color` (color)

The color of this grob.

`common-shortest-duration` (moment)

The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)

A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)

Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list of number pairs)

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`count-from` (integer)

The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`damping` (number)

Amount of beam slope damping.

`dash-definition` (pair)

List of dash-elements defining the dash structure. Each dash-element has a starting t value, an ending t-value, a dash-fraction, and a dash-period.

`dash-fraction` (number)

Size of the dashes, relative to dash-period. Should be between 0.1 and 1.0 (continuous line). If set to 0.0, a dotted line is produced

`dash-period` (number)

The length of one dash together with whitespace. If negative, no line is drawn at all.

`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`).

`denominator-style` (symbol)

The style of denominators in a time signature.

`details` (alist, with symbols as keys)

An alist of parameters for detailed grob behavior. See Section 3.1 [All layout objects], page 526, for more information on the available parameters and their default values of a particular grob's `details` property. See Section 3.2 [Graphical Object Interfaces], page 749,

for documentation of the available parameters. Supporting interfaces can be found at the bottom of a grob's description section.

`digit-names` (vector)

Names for string finger digits.

`direction` (direction)

If `side-axis` is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

`dot-count` (integer)

The number of dots.

`dot-negative-kern` (number)

The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)

List consisting of (*description string-number fret-number finger-number*) entries used to define fret diagrams.

`double-stem-separation` (number)

The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer)

The 2-log of the note head duration, i.e., 0 = whole note, 1 = half note, etc.

`eccentricity` (number)

How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)

A pair of numbers specifying the heights of the vertical edges: (*left-height . right-height*).

`edge-text` (pair)

A pair specifying the texts to be set at the edges: (*left-text . right-text*).

`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 flags to be displayed within markups (via `\note-by-number`). Available are ‘modern-straight-flag’, ‘old-straight-flag’, ‘flat-flag’, ‘mensural’, ‘stacked’, and ‘default’.

flat-positions (list)

Flats in key signatures are placed within the specified ranges of staff positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff position at which each clef places C: (alto treble tenor soprano baritone mezzosoprano bass). If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff position.

font-encoding (symbol)

The font encoding is the broadest category for selecting a font. Currently, only Lilypond’s system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

font-family (symbol)

The font family is the broadest category for selecting text fonts. Options include serif, sans and typewriter.

font-features (list)

Opentype features.

font-name (string)

This property is kept for backwards compatibility only. Use the `fonts` property instead.

font-series (symbol)

Select the series of a font. Common choices are normal and bold. The full list of symbols that can be used is: thin, ultralight (or extralight), light, semilight (or demilight), book, normal (or regular), medium, semibold (or demibold), bold, ultrabold (or extrabold), heavy (or black), and ultraheavy (or ultrablack or extrablack).

font-shape (symbol)

Select the shape of a font. Possible values are upright, italic, oblique, and slanted (which is the same as oblique).

**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.

**font-stretch (symbol)**

Select a condensed or expanded font, if available in the font family. Possible values are `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `normal`, `semi-expanded`, `expanded`, `extra-expanded`, and `ultra-expanded`.

**font-variant (symbol)**

Select the variant of a font. Choices include `normal` and `small-caps`.

**fonts (alist, with symbols as keys)**

An alist mapping font families to font names. The standard font families are `music`, `serif`, `sans` and `typewriter`.

**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 down-stem voice note; if there are no down-stem voices, the width of the first up-stem voice note is taken instead. This is used by Section “note-collision-interface” in *Internals Reference*.

**forced-spacing (number)**

Spacing forced between grobs, used in various ligature engravers.

**fraction (pair of numbers)**

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`.
  - `barre-thickness` – Thickness of barre line, in multiples of `dot-radius`. Only defined for `barre-type=straight`. Default value 1.
- `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-overhang` – Extension of string lines beyond last fret line, in multiples of fret-distance. Default value 1.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string  $k$  is given by  $\text{thickness} * (1 + \text{string-thickness-factor})^{(k-1)}$ . Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)

How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)

Run to the extent of the column for a full-length tuplet bracket.



`full-measure-extra-space` (number)

Extra space that is allocated at the beginning of a measure with only one note. This property is read from the `NonMusicalPaperColumn` that begins the measure.

`full-size-change` (boolean)

Don't make a change clef smaller.

`gap` (dimension, in staff space)

Size of a gap in a variable symbol.

`gap-count` (integer)

Number of gapped beams for tremolo.

`glissando-skip` (boolean)

Should this `NoteHead` be skipped by glissandi?

`glyph` (string)

A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-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 or clefs, *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 default 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. Default value 0.15.
- `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)

Are the note heads left or right in a semitie?

`height` (dimension, in staff space)

Height of an object in staff-space units.

`height-limit` (dimension, in staff space)

Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)

If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizon-padding` (number)

The amount to pad the axis along which a Skyline is built for the side-position-interface.

`horizontal-shift` (integer)

An integer that identifies ranking of 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-quant`s (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 kneed?

`knee-spacing-correction` (number)

Factor for the optical correction amount for kneed beams. Set between 0 for no correction and 1 for full correction.

`knee-to-beam` (boolean)

Determines whether a tuplet number will be positioned next to a kneed beam.

`labels` (list)

List of labels (symbols) placed on a column.

`layer` (integer)

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-extra` (dimension, in staff space)

Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)

The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)

Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`ledger-positions-function` (any type)

A quoted Scheme procedure that takes a `StaffSymbol` grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

`left-bound-info` (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-slope-factor` (non-negative number)

Factor for calculating the maximum tuplet bracket slope. Notice that there exists a homonymous property for slurs.

`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` (positive moment with no grace part)

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-X-space` (dimension, in staff space)

Minimum distance that the victim should move horizontally (after padding), overriding the `minimum-space` property value.

`minimum-Y-extent` (pair of numbers)

Minimum size of an object in Y dimension, measured in staff-space units.

`musical-length` (non-negative moment with no grace part)

Musical length.

`nested-fraction-mixed` (boolean)

Whether a fractional term of a time signature is printed as a mixed number (e.g., ‘2 1/2’) or as a common fraction (e.g., ‘5/2’).

`nested-fraction-orientation` (symbol)

A symbol describing the orientation of a fractional part of a time signature.

`nested-fraction-relative-font-size` (number)

The font size of a numeral in a fractional term of a time signature, relative to size of the whole numbers.

`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-dots-direction` (direction)

Whether the augmentation dots are shifted up or down (or not shifted) relative to the note head in a number-over-note time signature.

`note-flag-style` (symbol)

The style of the flags in a number-over-note time signature. See `flag-style`.

`note-head-style` (symbol)

The style of the note head in a number-over-note time signature. See Section “Note head styles” in *Notation Reference*.

`note-names` (vector)

Vector of strings containing names for easy-notation note heads.

`note-staff-position` (number)

The position of the note in a number-over-note time signature. See `staff-position`.

`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 (`<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.

`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

```
((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.

`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 not a number, align on the parent's reference point. 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 a bracket indicating non-arpeggio or optional material, the length of the horizontal edges.

range-collapse-threshold (non-negative, exact integer)

If the length of a volta range is greater than or equal to this threshold, print it with a dash. For example, if this is 3, a \volta 1,2,3 is printed as '1.-3.', but if it is 4, it is printed as '1.2.3.'.

rank-on-page (number)

0-based index of the system on a page.

ratio (number)

Parameter for slur shape. The higher this number, the quicker the slur attains its height-limit.

remove-empty (boolean)

If set, remove group if it contains no interesting items.



`remove-first` (boolean)

Remove the first staff of an orchestral score?

`remove-layer` (index or symbol)

When set as a positive integer, the `Keep_alive_together_engraver` removes all `VerticalAxisGroup` grobs with a `remove-layer` larger than the smallest retained `remove-layer`. Set to `#f` to make a layer independent of the `Keep_alive_together_engraver`. Set to `'()`, the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: `#'any` to keep the layer alive with any other layer in the group; `#'above` or `#'below` to keep the layer alive with the context immediately before or after it, respectively.

`replacement-alist` (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.

`rhythmic-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-justified` (boolean)

Used for `BarLines` to right-align them. Usually the extent of a `BarLine` has some positive value to the right. If this property is set to `#t`, `BarLine.stencil` is translated to the left by this value. Needs to be set at `Score` or `StaffGroup` level. As a result all `BarLines` of said `Score` or `StaffGroup` are right-justified.

`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. If not a number, align on the object's reference point.

`self-alignment-Y` (number)

Like `self-alignment-X` but for the Y axis.

`senza-misura-stencil` (stencil)

The symbol to print when `TimeSignature.fraction` is not set. Overriding `TimeSignature.stencil` circumvents this.

`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.

`short-bar-extent` (pair of numbers)

The Y-extent of a short bar line. The default is half the normal bar extent, rounded up to an integer number of staff spaces.

`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 `#t` 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)

If `space-alist` is defined for a grob that gets spaced in a staff, an entry for `first-note` must be present. If there is no `next-note` entry, the value of `first-note` is used instead.

Choices for *spacing-style* are:

`extra-space`

Put this much space between the two grobs. The space is stretchable and shrinkable 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 and shrinkable 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 and shrinkable.

`shrink-space`

Only compatible with `first-note` and `next-note`. Put this much space between the two grobs. The space is only shrinkable.

`semi-shrink-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 shrinkable.

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)
```

`span-all-note-heads` (boolean)

If true, tuplet brackets are printed spanning horizontally from the first to the last note head instead of covering only the stems.

`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.

For ties, there is a distinction between exact and inexact values: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

`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).

`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.

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 a desired tie configuration that overrides the default. *position* is the offset from the center of the staff in half staff-space units, 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.

There is a distinction between exact and inexact values for *position*: an exact value serves as a rough vertical offset that gets further tuned to make the tie avoid staff lines. An inexact value is taken as the precise vertical offset without further adjustments.

`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.

`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.

`visible-over-note-heads` (boolean)

This prints a tuplet bracket when the bracket is set to be over the note heads. This option can be combined with the default tuplet bracket visibility style and with `#'if-no-beam`.

`voiced-position` (number)

The staff position of a voiced Rest, negative if the rest has direction DOWN.

`volta-number-offset` (pair of numbers)

The offset of the volta number relative to the upper left corner of the volta bracket.

`when` (moment)

Global time step associated with this column.

`whiteout` (boolean-or-number)

If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`.

Usually `#f` by default. If `whiteout-color` is set, use this color instead of white for the background.

`whiteout-color` (color)

The background color used if property `whiteout` is set.

`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-alignment-extent` (pair of numbers)

If a grob wants to align itself on a `PaperColumn` grob that doesn't contain note heads, use this horizontal extent as a placeholder.

`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.

Note that many objects have special positioning considerations, which cause any setting of `X-offset` to be ignored or modified, even though the object supports the `self-alignment-interface` (page 815).

`X-padding` (dimension, in staff space)

Add this much extra space between objects that are next to each other horizontally, overriding the padding property value.

`X-positions` (pair of numbers)

Pair of X staff coordinates of a spanner in the form (*left . right*), where both *left* and *right* are in staff-space units of the current staff.

`Y-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.

Note that many objects have special positioning considerations, which cause any setting of Y-offset to be ignored or modified, even though the object supports the self-alignment-interface (page 815).

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 (*notename* . *groblist*) entries.

add-cauda (boolean)

Does this flexa require an additional cauda on the left side?

add-join (boolean)

Is this ligature head-joined with the next one by a vertical line?

add-stem (boolean)

Is this ligature head a virga and therefore needs an additional stem on the right side?

adjacent-pure-heights (pair)

A pair of vectors. Used by a VerticalAxisGroup to cache the Y-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.

allow-span-bar-above (boolean)

If false, no inter-staff bar line will be created above this item.

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-left` (array of grobs)

An array of bar line pointers for the left side of a volta bracket.

`bars-right` (array of grobs)

An array of bar line pointers for the right side of a volta bracket.

`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)

A pointer to the text grob of an analysis bracket.

`break-alignment` (graphical (layout) object)

The BreakAlignment (page 555), in a NonMusicalPaperColumn (page 655).

`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 deminished?

descendens (boolean)

Is this neume of descendent type?

direction-source (graphical (layout) object)

In case side-relative-direction is set, which grob to get the direction from.

dot (graphical (layout) object)

A reference to a Dots object.

dot-stencil (stencil)

The stencil for an individual dot, as opposed to a group of several dots.

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.

`french-beaming-stem-adjustment` (dimension, in staff space)

Stem will be shortened by this amount of space in case of French beaming style.

`glissando-index` (integer)

The index of a glissando in its note column.

`grace-spacing` (graphical (layout) object)

A run of grace notes.

`grob-defaults` (list)

A list with type and an alist of properties, derived from a list like `default-script-alist`, p.e.

```
(accent
 . ((avoid-slur . around)
 (padding . 0.20)
 (script-stencil . (feta . ("sforzato" . "sforzato")))
 (side-relative-direction . ,DOWN)))
```

`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.

`head-x-offset` (dimension, in staff space)

Extra horizontal offset for ligature heads.

`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-stencil` (stencil)

The stencil of a system's in-notes.

`in-note-system-padding` (number)

Padding between in-note and its associated system.

`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-down-stem` (boolean)

request a downward left stem for an initial breve in a ligature.

`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?

passage-direction (direction)

The placement of a passage-delimiter-interface item with respect to its passage: START at the start, END at the end, or CENTER otherwise.

pedal-text (graphical (layout) object)

A pointer to the text grob 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.

potential-beam (graphical (layout) object)

For tuplet brackets, a grob to use as parallel beam unless the tuplet is broken.

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-down-stem` (boolean)

request a downward right stem for a maxima in a ligature.

`right-items` (array of grobs)

Grobs organized on the right by a spacing object.

`right-neighbor` (graphical (layout) object)

See `left-neighbor`.

`right-up-stem` (boolean)

request an upward right stem for a final longa or maxima in a ligature.

`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.

strophæ (boolean)

Is this neume a strophæ?

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.

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?

volta-numbers (number list)

List of volta numbers.

X-common (graphical (layout) object)

Common reference point for axis group.

Y-common (graphical (layout) object)

See X-common.



## 4 Scheme functions

`add-bar-glyph-print-procedure` *glyph proc* [Function]

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* [Function]

Adds the given context *modification* to the list *contextmods* of context modifications.

`add-grace-property` *context-name grob sym val* [Function]

Set *sym*=*val* for *grob* in *context-name*.

`ly:add-interface` *iface desc props* [Function]

Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *callback disp cl* [Function]

Add the single-argument procedure *callback* as listener to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it calls *callback* with it.

`add-new-clef` *clef-name clef-glyph clef-position transposition c0-position* [Function]

Add a new clef to the list of supported clefs.

The arguments *clef-name*, *clef-glyph*, *clef-position*, and *transposition* are used to add an entry to the supported-clefs alist. Arguments *clef-glyph* and *c0-position* extend *c0-pitch-alist*.

`ly:add-option` *sym val description rest* [Function]

Add program option *sym* with default value *val* and docstring *description*.

LilyPond uses this function to define Scheme options available on the command line (given by `-d` or `--define-default`).

After start-up, command-line Scheme options are provided to LilyPond by function `ly:command-line-options`, which returns a key-value alist where all values are Scheme strings. Use the optional argument `#:type` to specify how such a value string for key *sym* should be processed by `ly:set-option`.

- If set to symbol string, don't do any further conversion and accept the value as a string. This is also necessary if a potentially fitting type predicate gets defined after LilyPond's command-line option handling (for example, `ly:duration?`). In such cases, type checking should be performed manually later on.
- If set to symbol string-or-boolean, do the same as with string but convert strings `"#f"` and `"#t"` to Boolean values.
- If set to symbol string-or-false, do the same as with string but convert a string value `"#f"` to Boolean value `#f`.
- If set to a procedure, handle the value as a Scheme expression and use the procedure as a predicate to check whether the value fits. This is also the default behaviour if `#:type` is not set, using `boolean?` as the procedure.
- If set to a list, handle the value as a Scheme expression and check whether it is one of the list's elements (using `equal?` for the comparison test).

'Handling as a Scheme expression' means that the string gets passed to the read Scheme function, which stops reading after the first complete Scheme expression has been parsed. As

a consequence, both strings "foo" and "foo bar" get converted to symbol foo, while a string "(foo" causes an error because the Scheme expression is not complete.

Passing #:internal? #t makes the option an internal option, not displayed in the lilypond -dhelp output (but displayed in lilypond -dhelp-internal).

Passing #:accumulative? #t makes the option accumulative, which gathers -d values in a list instead of letting the last -d flag overwrite the others.

`add-simple-time-signature-style` *style proc* [Function]  
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* [Function]  
Add a stroke glyph (from the music font) to the given flag stencil.  
This is an auxiliary function for `create-glyph-flag`.

`add-stroke-straight` *stencil grob dir log stroke-style offset length thickness stroke-thickness* [Function]  
Add an acciaccatura stroke to the given flag stencil.  
This is an auxiliary function for `straight-flag`.

`alist->hash-table` *lst* [Function]  
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` [Function]  
Return the hash table with all grob interface descriptions.

`ly:all-options` [Function]  
Get all option settings in an alist.

`ly:all-output-backend-commands` [Function]  
Return the list of extra output backend commands that are used internally in file `lily/stencil-interpret.cc`.

`ly:all-stencil-commands` [Function]  
Return the list of stencil commands that can be defined in the output modules (in files `output-*.scm`).

`ly:all-stencil-expressions` [Function]  
Return all symbols recognized as stencil expressions.

`allow-volta-hook` *bar-glyph* [Function]  
Allow the volta bracket hook being drawn over bar line *bar-glyph*.

`alterations-in-key` *pitch-list* [Function]  
Count number of sharps minus number of flats.

`ly:angle` *x y* [Function]  
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* [Function]  
Take *angle* (in radians) and map it between 0 and 2pi.

`angle-0-360` *angle* [Function]

Take *angle* (in degrees) and map it between 0 and 360 degrees.

`ly:append-to-option` *var val* [Function]

Add value *val* to an accumulative program option *var*.

See also function `ly:add-option`.

`array-copy/subarray!` *src dst offsets ...* [Function]

Similar to `array-copy`, but takes extra parameters for the start of a subarray where to copy. For example:

```
(let ((arr (make-array 'a 4 4))
 (to-copy (make-array 'b 2 2)))
 (array-copy/subarray! to-copy arr 2 1)
 arr)
⇒
#2((a a a a)
 (a a a a)
 (a b b a)
 (a b b a))
```

`arrow-stencil` *x y thick staff-space grob* [Function]

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?* [Function]

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?*.

`assert` ... [Macro]

Use `(assert condition)` or `(assert condition extra-failure-message)` to check that *condition* is true, and raise an error otherwise. Use this for conditions that should always be true, barring bugs; raise a more informative error if protecting against a user error.

`ly:assoc-get` *key alist default-value strict-checking* [Function]

Return value if *key* in *alist*, else *default-value* (or #f if not specified). If *strict-checking* is set to #t and *key* is not in *alist*, a programming error is output.

`assoc-get` - - [- [-]] [Function]

- 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.

`at-bar-line-substitute-caesura-type` *substitute-type* [Function]

At a bar line, create the caesura using *substitute-type* rather than the value of `caesuraType`.

`ly:axis-group-interface::add-element` *grob grob-element* [Function]

Add *grob-element* to the axis group *grob*. In particular, *grob* becomes parent to *grob-element* on all axes supported by *grob*, unless the parents are already set.

`ly:bar-line::calc-anchor` *grob* [Function]

Calculate the anchor position of a bar line. The anchor is used for the correct placement of bar numbers, etc.

- `bar-line::calc-break-visibility` *grob* [Function]  
Calculate the visibility of a bar line at line breaks.
- `bar-line::calc-glyph-name` *grob* [Function]  
Return the name of the bar line glyph printed by *grob* for the actual break direction.
- `bar-line::calc-glyph-name-for-direction` *glyphs dir* [Function]  
Find the glyph name for a bar line. *glyphs* is the list of bar-line types to consider in order. Each must have been defined with `define-bar-line`. *dir* is the break direction to consider: LEFT = end of line, CENTER = middle of line, RIGHT = start of line.
- `bar-line::compound-bar-line` *grob bar-glyph extent* [Function]  
Build the bar line stencil.
- `bar-line::draw-filled-box` *x-ext y-ext thickness extent grob* [Function]  
Return a straight bar line created by `ly:round-filled-box` looking at *x-ext*, *y-ext*, and *thickness*. The blot is calculated from *extent* and *grob*. *y-ext* is not necessarily equal to *extent*.
- `ly:bar-line::print` *grob* [Function]  
The print routine for bar lines.
- `bar-line::widen-bar-extent-on-span` *grob extent* [Function]  
Widen the bar line *extent* towards span bars adjacent to *grob*.
- `ly:base64-encode` *bv* [Function]  
Encode the given bytevector as a base 64 string.
- `ly:basic-progress` *str rest* [Function]  
A Scheme callable function to issue a basic progress message *str*. The message is formatted with *format*; *rest* holds the formatting arguments (if any).
- `beam-exceptions` *time-signature time-signature-settings* [Function]  
Get `beamExceptions` value for *time-signature* from *time-signature-settings*.
- `beat-base` *time-signature time-signature-settings* [Function]  
Get `beatBase` rational value for *time-signature* from *time-signature-settings*.
- `beat-structure` *beat-base time-signature time-signature-settings* [Function]  
Get `beatStructure` value in *beat-base* units for *time-signature* from *time-signature-settings*.
- `bend::arrow-head-stencil` *thickness x-y-coords height width dir* [Function]  
Return an arrow head stencil, calculated from the given dimensions *height* and *width*, and translated to *x-y-coords*, the end of the bend-spanners (curved) line.
- `bend::calc-bend-x-begin` *bend-spanner bounding-noteheads factor quarter-tone-diffs* [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]  
*top-right-tab-nhd*  
 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 'parenthesized of all relevant note heads of spanners right bound to #t. 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.
- `bit-list->byte-list` *bit-list* [Function]  
 Convert the given list of bits (booleans), whose length must be a multiple of 8, into a list of bytes (integers between 0 and 255).
- `bit-list->int` *bit-list* [Function]  
 Convert the given list of booleans to the number that it represents in binary.
- `ly:book?` *x* [Function]  
 Is *x* a smob of class `Book`?
- `ly:book-add-bookpart!` *book-smob book-part* [Function]  
 Add *book-part* to *book-smob* book part list.
- `ly:book-add-score!` *book-smob score* [Function]  
 Add *score* to *book-smob* score list.
- `ly:book-book-parts` *book* [Function]  
 Return book parts in *book*.
- `book-first-page` *layout props* [Function]  
 Return the 'first-page-number of the entire book.
- `ly:book-header` *book* [Function]  
 Return header in *book*.
- `ly:book-paper` *book* [Function]  
 Return paper in *book*.
- `ly:book-process` *book-smob default-paper default-layout output* [Function]  
 Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

- `ly:book-process-to-systems` *book-smob default-paper default-layout output* [Function]  
 Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).
- `ly:book-scores` *book* [Function]  
 Return scores in *book*.
- `ly:book-set-header!` *book module* [Function]  
 Set the book header.
- `box-grob-stencil` *grob* [Function]  
 Make a box of exactly the extents of the grob. The box precisely encloses the contents.
- `box-stencil` *stencil thickness padding* [Function]  
 Add a box around *stencil*, producing a new stencil.  
 The spacing characteristics are preserved if either a `\hspace` or a `\vspace` command (with either positive or negative values) gets boxed. This means that *padding* and *thickness* added to an empty extent will not participate in spacing and is not recognized by skylines.
- `ly:bp` *num* [Function]  
*num* bigpoints (1/72th inch).
- `ly:bracket` *a iv t p* [Function]  
 Make a bracket in direction *a*. The extent of the bracket is given by *iv*. The wings protrude by an amount of *p*, which may be negative. The thickness is given by *t*.
- `bracketify-stencil` *stil axis thick protrusion padding* [Function]  
 Add brackets around *stil*, producing a new stencil.
- `break-alignable-interface::self-alignment-of-anchor` *g* [Function]  
 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* [Function]  
 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` [Function]  
 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* [Function]  
 Return a callback that calculates a value based on a grob's break direction.
- `ly:broadcast` *disp ev* [Function]  
 Send the stream event *ev* to the dispatcher *disp*.
- `byte-list->bit-list` *byte-list* [Function]  
 Convert a list of bytes (integers between 0 and 255) into a list of bits (booleans).
- `caesura-script-interface::before-line-breaking` *script* [Function]  
 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.
- `calc-measure-length` *time-signature* [Function]  
 Calculate the measure length for *time-signature*.
- `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.

- `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*).
- `chord-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)
- `chord-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-modifier::print grob` [Function]  
 Callback for ClefModifier grob.
- `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` [Function]  
*num* cm.
- `collect-book-music-for-book book music` [Function]  
 Book music handler.
- `collect-bookpart-for-book book-part` [Function]  
 Top-level book-part handler.
- `collect-music-aux score-handler music` [Function]  
 Pass *music* to *score-handler*, with preprocessing for page layout instructions.
- `collect-music-for-book music` [Function]  
 Top-level music handler.
- `ly:command-line-code` [Function]  
 The Scheme code specified on the command line with option -e.
- `ly:command-line-options` [Function]  
 The Scheme options specified on the command line with option -d.  
 Return a key-value alist, with keys being symbols and values being strings.



- `comparator-from-key` *key cmp* [Function]  
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* [Function]  
Make the dispatcher *to* listen to events from *from*.
- `construct-chord-elements` *root duration modifications* [Function]  
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* [Function]  
Is *x* a smob of class Context?
- `ly:context-children` *context* [Function]  
Return a list with the children contexts of *context*.
- `ly:context-current-moment` *context* [Function]  
Return the current moment of *context*.
- `ly:context-def?` *x* [Function]  
Is *x* a smob of class Context\_def?
- `ly:context-def-lookup` *def sym val* [Function]  
Return the value of *sym* in context definition *def* (e.g., \Voice). If no value is found, return *val* or '()' if *val* is undefined. *sym* can be any of 'default-child', 'consists', 'description', 'aliases', 'accepts', 'property-ops', 'context-name', 'group-type'.
- `ly:context-def-modify` *def mod* [Function]  
Return the result of applying the context-mod *mod* to the context definition *def*. Does not change *def*.
- `ly:context-event-source` *context* [Function]  
Return event-source of *context*.
- `ly:context-events-below` *context* [Function]  
Return a stream-distributor that distributes all events from *context* and all its subcontexts.
- `ly:context-find` *context name* [Function]  
Find a context with name or alias *name*, first considering *context* and then searching its ancestors. Return #f if not found.
- `ly:context-grob-definition` *context name* [Function]  
Return the definition of *name* (a symbol) within *context* as an alist.
- `ly:context-id` *context* [Function]  
Return the ID string of *context*, i.e., for \context Voice = "one" ... return the string one.
- `ly:context-matched-pop-property` *context grob cell* [Function]  
This undoes a particular \override, \once \override or \once \revert when given the specific alist pair to undo.
- `ly:context-mod?` *x* [Function]  
Is *x* a smob of class Context\_mod?

- `ly:context-mod-apply! context mod` [Function]  
Apply the context modification *mod* to *context*.
- `ly:context-name context` [Function]  
Return the name of *context*, i.e., for `\context Voice = "one"` ... return the symbol `Voice`.
- `ly:context-output-def context` [Function]  
Return the output definition of *context*.
- `ly:context-parent context` [Function]  
Return the parent of *context*, #f if none.
- `ly:context-property context name rest` [Function]  
Get the value of property *name* visible in *context*. The first *rest* argument may optionally be an alternative value to return when the property value is '(). Following that, there may appear keyword options:
- `#:default`  
The value to return when the property is not set. When this option is absent, the same value is returned as when the property value is '().
  - `#:search-ancestors?`  
#f limits the search to *context*. The default is #t.
- `ly:context-property-pop context name` [Function]  
Remove the top entry from the stack for property *name* in context *context* and set or unset the property. If the stack is empty, throw `ly:context-property-stack-underflow` and do not change the property.
- `ly:context-property-push context name` [Function]  
Add the current value (or lack thereof) of property *name* in context *context* to a context-specific stack. The state should be restored later with a paired `ly:context-property-pop` for the same context and property.
- `ly:context-property-where-defined context name def` [Function]  
Return the context above *context* where property *name* is defined, or *def* (defaulting to '()) if no such context is found.
- `ly:context-pushpop-property context grob eltprop val` [Function]  
Do `\temporary \override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).
- `ly:context-schedule-moment context moment` [Function]  
Add *moment* (which must lie in the future) to the list of moments to process for the global context governing *context*. This makes it possible for translators (engravers, performers) to act at moments not directly created by user input.
- `ly:context-set-property! context name val` [Function]  
Set value of property *name* in *context* to *val*.
- `context-spec-music m context [id [mods]]` [Function]  
Add `\context context = id \with mods` to *m*.
- `ly:context-unset-property context name` [Function]  
Unset value of property *name* in *context*.

`copy-repeat-chord` *original-chord repeat-chord duration event-types* [Function]  
 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* [Function]  
 Given *lst* as (E1 E2 . . .), return ((E1 . 1) (E2 . 2) . . .).

`create-glyph-flag` *flag-style dir-modifier grob* [Function]  
 Create a flag stencil by looking up the glyph from the music font.  
 This is an auxiliary function for *mensural-flag*, *glyph-flag*, and *normal-flag*.

`cross-staff-connect` *stem* [Function]  
 Set cross-staff property of the stem to this function to connect it to other stems automatically

`cue-substitute` *quote-music* [Function]  
 Must happen after *quote-substitute*.

`cyclic-base-value` *value cycle* [Function]  
 Take *value* (for example, an angle) and modulo-maps it between 0 and base *cycle*.

`ly:debug` *str rest* [Function]  
 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* [Function]  
 Create a flag stencil for the stem.  
 The flag style is derived from the *style* property of *grob* (which must be of type *Flag*).  
 By default, LilyPond uses a C++ function (which is slightly faster) to do exactly the same as this function. However, if you want to modify the default flags this function can be used to obtain the default flag stencil, which can then be modified at will.  
 The available, predefined values for *style* are "" (empty, for normal flags), "mensural", and "no-flag". Other values are used to construct glyph names for flags; see function *glyph-flag* for details.  
 Example:

```
\override Flag.stencil = #default-flag
\override Flag.style = #'mensural
```

`ly:default-scale` [Function]  
 Get the global default scale.

`define-bar-line` *bar-glyph eol-glyph bol-glyph span-glyph* [Function]  
 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*).  
 After definition, *bar-glyph* is accepted as an argument to the `\bar` command. To distinguish bar lines with the same unbroken glyph, *bar-glyph* may be annotated with text after a hyphen, as in the predefined `\bar ".|:-|"` and `\bar ".|:-||"`.  
 The substitute glyphs may be either strings or booleans.  
 The value `#f` or the string "x" call for no glyph. Unlike `#f`, "x" may be annotated and may be used in *bar-glyph*, as in the predefined `\bar "x-."`. The empty string, "", calls for a zero-width stencil; it also may be annotated, as in the predefined `\bar "-span|"`.  
 The value `#t` calls for the same value as *bar-glyph*. Note that this includes any annotations, which can affect things like which span bar is chosen or whether a volta bracket closes.  
 See Section "List of bar lines" in *Notation Reference*.

`define-deprecated-property` *category-type-symbol deprecated-symbol* [Function]  
*deprecated-type? #:new-symbol new-symbol #:new->old new->old*  
*#:old->new old->new #:warning warning*

If *warning* is *#f*, a default warning will be generated.

`define-event-class` *class parent* [Function]  
 Defines a new event class derived from *parent*, a previously defined event class.

`define-event-function` ... [Macro]  
 Like `define-music-function`, but the return value must be a post-event.

`define-fonts` *paper define-font define-pango-pf* [Function]  
 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` ... [Macro]  
 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 `markup->string`, passing it *#:layout layout* *#:props props*.

The autogenerated documentation makes use of some optional specifications that are otherwise ignored:

- *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 *command-markup* instead of a (*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 *command-markup* function according to the parameters of the definition: *markup-command-signature*, *markup-function-category*, *markup-function-properties*.

`define-markup-list-command ...` [Macro]

Same as `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 `markup-list-function?` object property to `#t`.

`define-music-function ...` [Macro]

Define and return a music function. Syntax:

```
(define-music-function (arg1 arg2 ...)
 (type1? type2? ...)
 function-body)
```

*type1?*, *type2?*, etc., can take one of the forms *predicate?* for mandatory arguments satisfying the predicate, (*predicate?*) for optional parameters of that type defaulting to `#f`, (*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.

`define-scheme-function ...` [Macro]

Like `define-music-function`, but the return type is not restricted to music.

`define-syntax-function ...` [Macro]

Helper macro for `ly:make-music-function`. Syntax:

```
(define-syntax-function result-type?
 (arg1 arg2 ...)
 (type1? type2? ...)
 function-body)
```

See `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.

`define-tag-group tags` [Function]

Define a tag group consisting of the given *tags*, a list of symbols. Returns `#f` if successful, and an error message if there is a conflicting tag group definition.

`define-void-function ...` [Macro]

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` [Function]

Convert the given angle from degrees to radians.

- `descend-to-context` *m context [id [mods]]* [Function]  
Like `context-spec-music`, but only descending.
- `determine-split-list` *evl1 evl2 chord-range* [Function]  
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* [Function]  
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. If the context-property `supportNonIntegerFret` is set #t, micro-tones are supported for `TabStaff`, but not for `FretBoards`.
- `ly:dimension?` *d* [Function]  
Is *d* a dimension? Used to distinguish length variables from normal numbers.
- `ly:dir?` *s* [Function]  
Is *s* a direction? Valid directions are -1, 0, or 1, where -1 represents left or down, 1 represents right or up, and 0 represents a neutral direction.
- `dir-basename` *file rest ...* [Function]  
Strip suffixes in *rest*, but leave directory component for *file*.
- `ly:directed` *direction magnitude* [Function]  
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.
- `direction-scaled` *val* [Function]  
Returns *val*, scaled by 'direction of *grob*. The return value is used for extra-spacing-height to push note columns right in `markLengthOn`.
- `ly:disconnect-dispatchers` *to from* [Function]  
Stop the dispatcher *to* listening to events from *from*.
- `ly:dispatcher?` *x* [Function]  
Is *x* a smob of class `Dispatcher`?
- `display-lily-music` *expr [port]* [Function]  
Display the music expression *expr* using LilyPond syntax.
- `display-music` *music [port]* [Function]  
Display *music*, not done with `music-map` for clarity of presentation.
- `display-scheme-music` *obj [port]* [Function]  
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* [Function]  
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.

|                                                                                                                                                                                                                                                                                               |            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| <code>ly:duration? x</code>                                                                                                                                                                                                                                                                   | [Function] |
| Is <i>x</i> a smob of class <i>Duration</i> ?                                                                                                                                                                                                                                                 |            |
| <code>ly:duration&lt;? p1 p2</code>                                                                                                                                                                                                                                                           | [Function] |
| Is <i>p1</i> shorter than <i>p2</i> ?                                                                                                                                                                                                                                                         |            |
| <code>ly:duration-&gt;moment dur</code>                                                                                                                                                                                                                                                       | [Function] |
| Convert <i>dur</i> to a moment with no grace part.                                                                                                                                                                                                                                            |            |
| <code>ly:duration-&gt;number dur</code>                                                                                                                                                                                                                                                       | [Function] |
| Convert <i>dur</i> to the equivalent number of whole notes.                                                                                                                                                                                                                                   |            |
| <code>ly:duration-&gt;string dur</code>                                                                                                                                                                                                                                                       | [Function] |
| Convert <i>dur</i> to a string.                                                                                                                                                                                                                                                               |            |
| <code>ly:duration-compress dur factor</code>                                                                                                                                                                                                                                                  | [Function] |
| Compress <i>dur</i> by rational <i>factor</i> .                                                                                                                                                                                                                                               |            |
| <code>ly:duration-dot-count dur</code>                                                                                                                                                                                                                                                        | [Function] |
| Extract the dot count from <i>dur</i> .                                                                                                                                                                                                                                                       |            |
| <code>duration-dot-factor dotcount</code>                                                                                                                                                                                                                                                     | [Function] |
| Given a count of the dots used to extend a musical duration, return the numeric factor by which they increase the duration.                                                                                                                                                                   |            |
| <code>ly:duration-factor dur</code>                                                                                                                                                                                                                                                           | [Function] |
| Extract the compression factor from <i>dur</i> . Return it as a pair.                                                                                                                                                                                                                         |            |
| <code>duration-line::calc grob</code>                                                                                                                                                                                                                                                         | [Function] |
| Return list of values needed to print a stencil for <i>DurationLine</i> .                                                                                                                                                                                                                     |            |
| <code>duration-line::print grob</code>                                                                                                                                                                                                                                                        | [Function] |
| Return the stencil of <i>DurationLine</i> .                                                                                                                                                                                                                                                   |            |
| <code>ly:duration-log dur</code>                                                                                                                                                                                                                                                              | [Function] |
| Extract the duration log from <i>dur</i> .                                                                                                                                                                                                                                                    |            |
| <code>duration-log-factor lognum</code>                                                                                                                                                                                                                                                       | [Function] |
| Given a logarithmic duration number, return the length of the duration, as a number of whole notes.                                                                                                                                                                                           |            |
| <code>ly:duration-scale dur</code>                                                                                                                                                                                                                                                            | [Function] |
| Extract the compression factor from <i>dur</i> . Return it as a rational.                                                                                                                                                                                                                     |            |
| <code>duration-visual dur</code>                                                                                                                                                                                                                                                              | [Function] |
| 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.                                                                                                                            |            |
| <code>duration-visual-length dur</code>                                                                                                                                                                                                                                                       | [Function] |
| 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.                                                                                                                                                   |            |
| <code>dynamic-text-spanner::before-line-breaking grob</code>                                                                                                                                                                                                                                  | [Function] |
| Monitor left bound of <i>DynamicTextSpanner</i> for absolute dynamics. If found, ensure <i>DynamicText</i> does not collide with spanner text by changing 'attach-dir and 'padding. Reads the 'right-padding property of <i>DynamicText</i> to fine-tune space between the two text elements. |            |

`ly:effective-prefix` [Function]

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?` [Function]

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 *coords* 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.

```
#(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` [Function]

Add an ellipse around *stencil*, padded by the padding pair, producing a new stencil.

`end-broken-spanner? spanner` [Function]

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` [Function]

Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.

`ly:engraver-make-grob engraver grob-name cause` [Function]

Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.

`ly:engraver-make-item engraver grob-name cause` [Function]

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` [Function]

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` [Function]

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* [Function]  
 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* ... [Function]  
 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* [Function]  
 Is *obj* a proper (non-rhythmic) Event object?
- `event-chord-notes` *event-chord* [Function]  
 Return a list of all notes from *event-chord*.
- `event-chord-pitches` *event-chord* [Function]  
 Return a list of all pitches from *event-chord*.
- `event-chord-reduce` *music* [Function]  
 Reduce event chords in *music* to their first note event, retaining only the chord articulations. Returns the modified music.
- `event-chord-wrap!` *music* [Function]  
 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* [Function]  
 Copy *m* and all sub-expressions of *m*.
- `event-has-articulation?` *event-type stream-event* [Function]  
 Is *event-type* in the articulations list of the music causing *stream-event*?
- `ly:event-length` *event moment* [Function]  
 Return the length of a stream event. If *moment* is not given, this is just the event’s length property. If *moment* is given and is an in-grace moment (i.e. having non-zero, usually negative, grace part), then the length of the stream event is returned as a grace-only moment. In any case, thus, the effective length of the stream event when happening at *moment* is returned.
- `ly:event-property` *sev sym val* [Function]  
 Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or ‘()’ if *val* is not specified.
- `ly:event-set-property!` *ev sym val* [Function]  
 Set property *sym* in event *ev* to *val*.
- `expand-repeat-chords!` *event-types music* [Function]  
 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* [Function]  
 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` *str rest* [Function]  
 Register *str* as an expected warning that should be suppressed.  
 If this Scheme-callable function does not encounter the given warning, a warning about the missing warning is shown. If the same warning occurs multiple times, this function must be called multiple times, too.

*str* should usually be tagged as being translated with (G\_ ...), and *rest* holds the expected parameter values.

Example:

```
(ly:expect-warning (G_ "Bar number is ~a; expected ~a") 3 15)
```

`extract-beam-exceptions` *music* [Function]

Create a value useful for setting beamExceptions from *music*.

`extract-music` *music pred?* [Function]

Return a flat list of all music matching *pred?* inside of *music*, not recursing into matches themselves.

`extract-named-music` *music music-name* [Function]

Return a flat list of all music named *music-name* (either a single event symbol or a list of alternatives) inside of *music*, not recursing into matches themselves.

`ly:extract-subfont-from-collection` *collection-file-name idx subfont-file-name* [Function]

Extract the subfont of index *idx* in TrueType collection (TTC) or OpenType/CFF collection (OTC) file *collection-file-name* and write it to file *subfont-file-name*.

`extract-typed-music` *music type* [Function]

Return a flat list of all music with *type* (either a single type symbol or a list of alternatives) inside of *music*, not recursing into matches themselves.

`figured-bass-continuation::print` *grob* [Function]

Callback for BassFigureContinuation grobs.

`ly:find-file` *name strict* [Function]

Return the absolute file name of *name*. By default, if the file is not found, return #f. If the optional parameter *strict* is passed as #t, raise an error in this case instead.

`find-named-props` *prop-name grob-descriptions* [Function]

Used by \magnifyMusic and \magnifyStaff. If *grob-descriptions* is equal to the all-grob-descriptions alist (defined in scm/define-grobs.scm), this finds all grobs that can have a value for the *prop-name* property, and return them as a list in the following format:

```
'((grob prop-name)
 (grob prop-name)
 ...)
```

`find-pitch-entry` *keysig pitch accept-global accept-local* [Function]

Return the first entry in *keysig* that matches *pitch* by notename and octave. Alteration is not considered. *accept-global* states whether key signature entries should be included. *accept-local* states whether local accidentals should be included. If no matching entry is found, #f is returned.

`finger-glide::print` *grob* [Function]

The stencil printing procedure for grob FingerGlideSpanner. Depending on the grob property style several forms of appearance are printed. Possible settings for grob property style are zigzag, trill, dashed-line, dotted-line, stub-left, stub-right, stub-both, bow, none and line, which is the default.

`first-assoc` *keys lst* [Function]

Return first successful assoc of key from *keys* in *lst*.

- `first-broken-spanner?` *spanner* [Function]  
Is *spanner* broken *and* the first of its broken siblings? See also `unbroken-or-first-broken-spanner?`.
- `first-member` *members lst* [Function]  
Return first successful member (of member) from *members* in *lst*.
- `flat-flag` *grob* [Function]  
A callback function for `Flag.stencil` to get a flat flag.  
The up-stem and down-stem angles of the flags are both 0 degrees. If the caller sets the `stroke-style` property of *grob* to the string "grace", add a slash through the flag.  
This function returns a stencil.
- `flat-zip-longest` *lsts ...* [Function]  
Return a list made of the first element from the first list, then the first element from the second list, ..., the second element from the first list, ..., until all lists are exhausted. For example:  
$$(\text{flat-zip-longest } '(a\ b\ c\ d)\ '(e\ f)\ '(g\ h\ i)) \Rightarrow '(a\ e\ g\ b\ f\ h\ c\ i\ d)$$
- `flatten-list` *x* [Function]  
Unnest list.
- `flip-stencil` *axis stil* [Function]  
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* [Function]  
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’.
- `fold-values` *proc lst inits ...* [Function]  
A variant of `fold` that works on one list only, but allows *proc* to return multiple values, and can itself return multiple values. The calls to *proc* are *(proc list-elem previous1 previous2 ...)*. Note that the *inits* arguments are given after *lst* in the signature, unlike `fold`.
- `ly:font-config-add-directory` *dir* [Function]  
Add directory *dir* to `FontConfig`.
- `ly:font-config-add-font` *font* [Function]  
Add font *font* to `FontConfig`.
- `ly:font-config-display-fonts` *port* [Function]  
List all fonts visible to `FontConfig`, together with directory information.  
Optional argument *port* selects the output port; the default is `(current-error-port)`.
- `ly:font-config-get-font-file` *name* [Function]  
Get the file for font *name*, as found by `FontConfig`.
- `ly:font-design-size` *font* [Function]  
Given the font metric *font*, return the design size, relative to the current output-scale.

- `ly:font-file-name font` [Function]  
Given the font metric *font*, return the corresponding file name.
- `ly:font-get-glyph font name` [Function]  
Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.  
Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.
- `ly:font-glyph-name-to-index font name` [Function]  
Return the index for *name* in *font*.  
Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.
- `ly:font-index-to-charcode font index` [Function]  
Return the character code for *index* in *font*.  
Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.
- `ly:font-magnification font` [Function]  
Given the font metric *font*, return the magnification, relative to the current output-scale.
- `ly:font-metric? x` [Function]  
Is *x* a 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.
- `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 provided 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` *grob common* [Function]
 Determine the extent of *grob* relative to *common* along the x axis, finding its extent as a bound when it has bound-alignment-interfaces property list set and otherwise the full extent.
- `ly:get-all-function-documentation` [Function]
 Get a hash table with all LilyPond Scheme extension functions.
- `ly:get-all-translators` [Function]
 Return a list of all translator objects that may be instantiated.
- `get-bound-note-heads` *spanner* [Function]
 Take a spanner grob and return a pair containing all note heads of the initial starting and the final NoteColumn.
- `ly:get-cff-offset` *font-file-name idx* [Function]
 Get the offset of the 'CFF' table for *font-file-name*, returning it as an integer. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.
- `get-chord-shape` *shape-code tuning base-chord-shapes* [Function]
 Return the chord shape associated with *shape-code* and *tuning* in the hash-table *base-chord-shapes*.
- `ly:get-context-mods` *contextmod* [Function]
 Returns the list of context modifications stored in *contextmod*.
- `ly:get-font-format` *font-file-name idx* [Function]
 Get the font format for *font-file-name*, returning it as a symbol. The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.
- `ly:get-option` *var* [Function]
 Get a global option setting.
- `get-postscript-bbox` *string* [Function]
 Extract the bounding box from *string*, or return #f if not present.
- `ly:get-spacing-spec` *from-scm to-scm* [Function]
 Return the spacing spec going between the two given grobs, *from-scm* and *to-scm*.
- `get-tweakable-music` *mus* [Function]
 When tweaking music, return a list of music expressions where the tweaks should be applied. Relevant for music wrappers and event chords.

`glyph-flag` *flag-style* [Function]

A callback for function `default-flag` to get a flag glyph.

This function actually constructs a function returning a stencil, expecting a single argument, *grob*.

It looks up `glyph flags.StyleDirLog` in the music font and uses it for the flag stencil. *Style* is the flag style based on *flag-style* (which can be empty), *Dir* is the flag direction (either ‘u’ or ‘d’), and *Log* the duration log (an integer in the range 3 to 10) from which the number of flags attached to the stem is derived. Both *Dir* and *Log* are taken from *grob*. Example: `flags.u3`.

If *grob* has the `stroke-style` property set, add a second glyph with the same glyph name components but use its value instead for *log*. Example: `flags.ugrace`.

Not to be used with mensural flags, which have a slightly different naming scheme (see function `mensural-flag`).

`ly:grob?` *x* [Function]

Is *x* a smob of class `Grob`?

`grob::all-objects` *grob* [Function]

Return a list of the names and contents of all properties having type `ly:grob?` or `ly:grob-array?` for all interfaces supported by *grob* *grob*.

`grob::compose-function` *func data* [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::directional-value` *value-when-negative value-when-positive*
`#:controlling-property controlling-property #:default default` [Function]

grob callback generator for returning a value depending on the sign of the numeric property identified by the `#:controlling-property` argument.

Values for negative and positive cases are required. The `#:default` argument specifies the return value when the controlling property is zero.

`grob::display-objects` *grob* [Function]

Display all objects stored in properties of *grob* *grob*.

`grob::inherit-parent-property` *axis property default* ... [Function]

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* [Function]

Return the name of the *grob* *grob* as a symbol.

`grob::offset-function` *func data rest* ... [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 *func* accepts a *grob* and returns a value that is added to the value resulting from *data*. Optional argument *plus* defaults to ‘+’ but may be changed to allow for using a different underlying accumulation.

If *data* is `#f` or ‘()’, it is not included in the sum.

- `grob::relay-directional-property` *negative-property* [Function]
positive-property *#:controlling-property* *controlling-property* *#:default*
default *#:default-property* *default-property*
grob callback generator for returning the value of another property depending on the sign of the numeric property identified by the *#:controlling-property* argument.
The *#:default* argument specifies the value to return when any lookup fails.
Property names for negative and positive cases are required. The optional *#:default-property* argument identifies a property to relay when the controlling value is zero; if none is provided, the default value is returned.
- `grob::relay-other-property` *property* [Function]
grob callback generator for returning the value of another property, which is identified by the symbol *property*.
- `grob::rhythmic-location` *grob* [Function]
Return a pair consisting of the measure number and moment within the measure of *grob*.
- `grob::unpure-Y-extent-from-stencil` *pure-function* [Function]
The unpure height will come from a stencil whereas the pure height will come from *pure-function*.
- `grob::when` *grob* [Function]
Return the global timestep (a Moment) of *grob*.
- `ly:grob-alist-chain` *grob* *global* [Function]
Get an alist chain for *grob*, with *global* as the global default. If unspecified, there is no global default.
- `ly:grob-array?` *x* [Function]
Is *x* a smob of class *Grob_array*?
- `ly:grob-array->list` *grob-arr* [Function]
Return the elements of *grob-arr* as a Scheme list.
- `ly:grob-array-length` *grob-arr* [Function]
Return the length of *grob-arr*.
- `ly:grob-array-ref` *grob-arr* *index* [Function]
Retrieve the *index*th element of *grob-arr*.
- `ly:grob-basic-properties` *grob* [Function]
Get the immutable properties of *grob*.
- `ly:grob-chain-callback` *grob* *proc* *sym* [Function]
Find the callback that is stored as property *sym* of *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.
- `ly:grob-common-refpoint` *grob* *other* *axis* [Function]
Find the common refpoint of *grob* and *other* for *axis*.
- `ly:grob-common-refpoint-of-array` *grob* *others* *axis* [Function]
Find the common refpoint of *grob* and *others* (a *grob-array*) for *axis*.
- `ly:grob-default-font` *grob* [Function]
Return the default font for *grob*.

- `ly:grob-extent grob refp axis` [Function]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.
- `ly:grob-get-vertical-axis-group-index grob` [Function]
Get the index of the vertical axis group the grob *grob* belongs to; return -1 if none is found.
- `ly:grob-interfaces grob` [Function]
Return the interfaces list of grob *grob*.
- `ly:grob-layout grob` [Function]
Get \layout definition from grob *grob*.
- `ly:grob-list->grob-array grob-list` [Function]
Convert a Scheme list of grobs to a grob array.
- `ly:grob-object grob sym val` [Function]
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` [Function]
Return the unbroken original grob of *grob*, *grob* may be an item or spanner.
- `ly:grob-parent grob axis def` [Function]
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` [Function]
Compare two grob priority queue entries. This is an internal function.
- `ly:grob-properties? x` [Function]
Is *x* a smob of class `Grob_properties?`
- `ly:grob-property grob sym val` [Function]
Return the value for property *sym* of *grob*. If no value is found, return *val* or '() if *val* is not specified.
- `ly:grob-property-data grob sym` [Function]
Return the value for property *sym* of *grob*, but do not process callbacks.
- `ly:grob-pure-height grob refp beg end val` [Function]
Return the pure height of *grob* given reffpoint *refp*. If no value is found, return *val* or '() if *val* is not specified.
- `ly:grob-pure-property grob sym beg end val` [Function]
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or '() if *val* is not specified.
- `ly:grob-pure-relative-coordinate grob refp start end` [Function]
Return the pure vertical coordinate of *grob* relative to *refp* between *start* and *end*.
- `ly:grob-relative-coordinate grob refp axis` [Function]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

- `ly:grob-robust-relative-extent` *grob refp axis* [Function]
 Get the extent in *axis* direction of *grob* relative to the *grob refp*, or (0,0) if empty.
- `ly:grob-script-priority-less` *a b* [Function]
 Compare two grobs by script priority. For internal use.
- `ly:grob-set-nested-property!` *grob symlist val* [Function]
 Set nested property *symlist* in grob *grob* to value *val*.
- `ly:grob-set-object!` *grob sym val* [Function]
 Set *sym* in grob *grob* to value *val*.
- `ly:grob-set-parent!` *grob axis parent-grob* [Function]
 Set *parent-grob* as the parent of grob *grob* in axis *axis*.
- `ly:grob-set-property!` *grob sym val* [Function]
 Set *sym* in grob *grob* to value *val*.
- `ly:grob-spanned-column-rank-interval` *grob* [Function]
 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* [Function]
 Return the y position of *sg* relative to the staff.
- `ly:grob-suicide!` *grob* [Function]
 Kill *grob*.
- `ly:grob-system` *grob* [Function]
 Return the system grob of *grob*.
- `grob-transformer` *property func* [Function]
 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* [Function]
 Translate *grob* on axis *a* over distance *d*.
- `ly:grob-vertical<?` *a b* [Function]
 Does *a* lie above *b* on the page?
- `group-into-ranges` *lst* [Function]
 Turn a (possibly unsorted) list of integers into a sorted list of ranges, represented as pairs. For example:

$$(\text{group-into-ranges } '(1\ 4\ 3\ 6\ 7\ 2)) \Rightarrow ((1\ .\ 4)\ (6\ .\ 7))$$
- `ly:gulp-file` *filename* [*size*] [Function]
 Same as `ly:gulp-file-utf8`, but decode the file as Latin 1. Warning: this is rarely what you want; consider using `ly:gulp-file-utf8` instead.
- `ly:gulp-file-utf8` *filename* [*size*] [Function]
 Find a file on the search path (with `ly:find-file`), and return its contents decoded as UTF-8. Raise an error if the file is not found.
 If the optional argument *size* is given, read at most *size* characters (*not* bytes) from the file.

`ly:has-glyph-names? font-file-name idx` [Function]
 Does the font for *font-file-name* have glyph names? The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:hash-table-keys tab` [Function]
 Return a list of keys in *tab*.

`headers-property-alist-chain headers` [Function]
 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` [Function]
 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.

`horizontal-script::calc-staff-position val` [Function]
 Set staff-position for horizontal script definitions in the alist for context property `scriptDefinitions`. This opens the possibility to set Y-offset or staff-position by override or tweak. Directly setting Y-offset in script definitions blocks this. *val* should usually be zero, but can be any other number in order to insert some offset from zero in y-axis direction.

`ly:in-event-class? ev cl` [Function]
 Does event *ev* belong to event class *cl*?

`ly:inch num` [Function]
num inches.

`index-map f lsts ...` [Function]
 Applies *f* to corresponding elements of *lists*, just as `map`, providing an additional counter starting at zero. *f* needs to have the counter in its arguments. For example:

```
(index-map (lambda (i elt)
  (format #f "~s is the element at index ~a" elt i))
  '(a b c d e))
```

`ly:input-both-locations sip` [Function]
 Return input location in *sip* as
 (file-name first-line first-column last-line last-column)

`ly:input-file-line-char-column sip` [Function]
 Return input location in *sip* as (file-name line char column).

`ly:input-location? x` [Function]
 Is *x* a smob of class `Input`?

`ly:input-message sip msg rest` [Function]
 Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:input-warning sip msg rest` [Function]
 Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

- `int->bit-list` *n* [*pad-length*] [Function]
 Return the representation of *n* in binary, as a list of booleans.
 If the optional argument *pad-length* is given, the list is padded with leading zeros to make it at least this long.
- `interpret-markup` - - - [Function]
 - LilyPond procedure: `ly:text-interface::interpret-markup`
 Convert a text markup into a stencil. *layout* is a `\\layout` block. *props* is an alist chain, i.e., a list of alists. *markup* is the markup text to be processed. See also `grob-interpret-markup`.
- `ly:interpret-music-expression` *mus* *ctx* [Function]
 Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.
- `interval-center` *x* [Function]
 Center the number pair *x*, if an interval.
- `interval-index` *interval* *dir* [Function]
 Interpolate *interval* between between left (*dir*=-1) and right (*dir*=+1).
- `interval-length` *x* [Function]
 Length of the number pair *x*, if an interval.
- `ly:intlog2` *d* [Function]
 The 2-logarithm of 1/*d*.
- `invalidate-alterations` *context* [Function]
 Invalidate alterations in *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 *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* [Function]
 Is *g* an Item object?
- `item::extra-spacing-height-including-staff` *grob* [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* [Function]
 The break status direction of item *it*. -1 means end of line, 0 unbroken, and 1 beginning of line.
- `ly:item-get-column` *it* [Function]
 Return the PaperColumn or NonMusicalPaperColumn associated with this Item.
- `ly:iterator?` *x* [Function]
 Is *x* a smob of class `Music_iterator`?
- `layout-line-thickness` *grob* [Function]
 Get the line thickness of the *grob*'s corresponding layout.
- `layout-set-absolute-staff-size` *sz* [Function]
 Set the absolute staff size inside of a `\\layout{}` block. *sz* is in points.

- `layout-set-staff-size` *sz* [Function]
Set the staff size inside of a `\layout{}` block. *sz* is in points.
- `left-align-at-split-notes` *grob* [Function]
Left-align LyricText if the parent NoteHead is split by Completion_heads_engraver
- `ly:length` *x y* [Function]
Calculate magnitude of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.
- `ly:lily-lexer?` *x* [Function]
Is *x* a smob of class Lily_lexer?
- `ly:lily-parser?` *x* [Function]
Is *x* a smob of class Lily_parser?
- `lilypond-main` *files* [Function]
Entry point for LilyPond.
- `lilypond-version-outdated?` *file-version lily-version* [Function]
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* [Function]
Make a line using layout information from grob *grob*.
- `list-insert-separator` *lst between* [Function]
Create new list, inserting *between* between elements of *lst*.
- `list-join` *lst intermediate* [Function]
Put *intermediate* between all elements of *lst*.
- `list-pad-left` *lst len filler fillers ...* [Function]
Same as `list-pad-right`, but add padding on the left.
- `list-pad-right` *lst len filler fillers ...* [Function]
Pad *lst* on the right by appending elements until its length is at least *len*. The elements are taken from the variadic arguments. For example:

$$(\text{list-pad-right } '(a\ b\ c)\ 10\ 'd\ 'e) \\ \Rightarrow (a\ b\ c\ d\ e\ d\ e\ d\ e\ d)$$
- `ly:listened-event-class?` *disp cl* [Function]
Does *disp* listen to any event type in the list *cl*?
- `ly:listened-event-types` *disp* [Function]
Return a list of all event types that *disp* listens to.
- `ly:listener?` *x* [Function]
Is *x* a smob of class Listener?
- `lookup-markup-command` *code* [Function]
Return (*function . signature*) for a markup command *code*, or return `#f`.

- `lyric-hyphen::vaticana-style grob` [Function]
 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` [Function]
 Allow interpretation of tildes as lyric tying marks.
- `make-accidental-dodecaphonic-rule octaveness laziness` [Function]
 Variation on function `make-accidental-rule` that creates an dodecaphonic accidental rule.
- `make-accidental-rule octaveness laziness` [Function]
 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 paper header scores` [Function]
 Make a \book of *paper* and *header* (which may be #f as well) containing \scores.
- `ly:make-book-part scores` [Function]
 Make a \bookpart containing \scores.
- `make-bow-stencil start stop thickness angularity bow-height orientation` [Function]
 Create a bow stencil. It starts at point *start*, ends at point *stop*. *thickness* is the thickness of the bow. The higher the value of number *angularity*, the more angular the shape of the bow. *bow-height* determines the height of the bow. *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 fraction` [Function]
 Make markup for the 'C' time signature style.
- `make-circle-stencil radius thickness fill` [Function]
 Make a circle of radius *radius* and thickness *thickness*.
- `make-clef-set clef-name` [Function]
 Generate the clef setting commands for a clef named *clef-name*.
- `make-connected-line points grob` [Function]
 Take a list of points, *points*. Return a line connecting *points*, using `ly:line-interface::line` and getting layout information from *grob*.
- `make-connected-path-stencil pointlist thickness x-scale y-scale connect fill` [Function]
 Make a connected path described by the list *pointlist*, beginning at point (0, 0), with thickness *thickness*, and scaled by *x-scale* in the x direction and *y-scale* in the y direction. *connect* and *fill* are boolean arguments that specify whether the path should be connected or filled, respectively.

- `ly:make-context-mod mod-list` [Function]
 Create a context modification, optionally initialized via the list of modifications *mod-list*.
- `make-cue-clef-set clef-name` [Function]
 Generate the clef setting commands for a cue clef with name *clef-name*.
- `make-cue-clef-unset` [Function]
 Reset the clef settings for a cue clef.
- `ly:make-dispatcher` [Function]
 Return a newly created dispatcher.
- `ly:make-duration length dotcount num den` [Function]
 Make a duration. *length* is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.
 The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.
 A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.
- `make-duration-of-length moment` [Function]
 Make duration of the given moment length.
- `make-ellipse-stencil x-radius y-radius thickness fill` [Function]
 Make an ellipse of x radius *x-radius*, y radius *y-radius*, and thickness *thickness* with fill defined by *fill*.
- `make-engraver ...` [Macro]
 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` [Function]
 Make a filled box.
- `ly:make-global-context output-def` [Function]
 Set up a global interpretation context, using the output block *output-def*. The context is returned.
- `ly:make-global-translator global` [Function]
 Create a translator group and connect it to the global context *global*. The translator group is returned.
- `make-glyph-time-signature-markup style fraction` [Function]
 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` [Function]
 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` [Function]
 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* [Function]
 Revert the grob property *gprop* for *grob*.
- `make-grob-property-set` *grob gprop val* [Function]
 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* [Function]
 Convert music variable *mus* to harmonics.
- `make-line-stencil` *width startx starty endx endy* [Function]
 Make a line stencil of given line width and set its extents accordingly.
- `ly:make-listener` *callback* [Function]
 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* [Function]
 Wrapper function for inverter-factory.
- `make-modal-transposer` *from to scale* [Function]
 Wrapper function for transposer-factory.
- `ly:make-moment` *m g gn gd* [Function]
 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* [Function]
 Make a C++ Music object and initialize it with *props*.
 This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.
- `make-music` *name music-properties* ... [Function]
 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* [Function]
 Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*’s cdr is a list containing either `ly:music?` predicates or other type predicates. Its car is the syntax function to call.

- `ly:make-music-relative!` *music pitch* [Function]
 Make *music* relative to *pitch*, return final pitch.
- `ly:make-output-def` [Function]
 Make an output definition.
- `make-oval-stencil` *x-radius y-radius thickness fill* [Function]
 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* [Function]
 Return page marker with label *label*.
- `ly:make-page-permission-marker` *symbol permission* [Function]
 Return page marker with page breaking and turning permissions.
- `ly:make-paper-outputter` *port alist default-callback* [Function]
 Create an outputter dumping to *port*. *alist* 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` *state-machine split-list* [Function]
 Generate a sequence of part combiner context changes from a split list.
- `make-part-combine-marks` *split-list [get-next-state]* [Function]
 Generate a sequence of segments alternating skips and part-combine-events for each active voice. There may be an initial skip that is not specced to any Voice context.
- `make-partial-ellipse-stencil` *x-radius y-radius start-angle end-angle* [Function]
thick connect fill
 Create an elliptical arc. *x-radius* is the x radius of the arc. *y-radius* is the y radius of the arc. *start-angle* is the starting angle of the arc (in degrees). *end-angle* is the ending angle of the arc (in degrees). *thick* is the thickness of the line. *connect* is a boolean flag indicating whether the end should be connected to the start by a line. *fill* is a boolean flag indicating whether the shape should be filled.
- `make-path-stencil` *path thickness x-scale y-scale fill #:line-cap-style* [Function]
line-cap-style #:line-join-style line-join-style
 Make a stencil based on the path described by the list *path*, with thickness *thickness*, and scaled by *x-scale* in the x direction and *y-scale* in the y direction (the difference with scaling the resulting stencil using `ly:stencil-scale` is that this scaling does not change the thickness). *fill* is a boolean argument that specifies whether the path should be filled. Valid path commands are
`moveto rmoveto lineto rlineto curveto rcurveto closepath`
 and their standard SVG single-letter equivalents
`M m L l C c Z z`
- `make-performer` ... [Macro]
 Like `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` *octave note alter* [Function]
 Make a pitch. *octave* is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob type init rest` [Function]
 Create a Prob object.

`ly:make-rand-seed` [Function]
 Create seed value for initialization of a pseudo-random generator by combining output from a high-resolution clock with the current process id.

`ly:make-regex pattern` [Function]
 Construct a new regular expression object.

Note that regular expressions created with this function are distinct from Guile native regular expressions (the latter don't fully support Unicode). They should be used with `ly:regex-...` functions.

The full reference for the supported regular expression syntax can be read at <https://www.pcre.org/original/doc/html/pcpattern.html>.

`make-relative ...` [Macro]

The list of pitch or music variables in *variables* is (when inside of a '`\relative`' expression) first passed through the throwaway expression *reference* for the sake of adjusting the variables according to the needs of relative notation, and then is employed for constructing the returned expression *music*.

This should work well both inside and outside of `\relative` even when music function arguments get used multiple times and/or in different order in the resulting music expression.

Outside of `\relative`, the result just reflects plugging in the *variables* into *music*.

Inside of `\relative`, however, `\relative` is getting called on the *reference* expression (that is supposed to contain the variables just once and in the order and arrangement that results in a natural action of `\relative` on their values). After adjusting the octaves in the variables in that manner, the resulting expression *music* is constructed from them.

Any of the *variables* containing a pitch rather than a complete music expression is replaced with a simple note event for the purpose of plugging into *reference* and thus is also affected by `\relative`.

For `\relative` to have an effect on one of the *variables*, the *reference* expression must use the values of the variables without creating copies (i.e., only using '`#`' instead of '`$`' on them inside of '`#{...#}`' constructs). The reference expression will usually just be a sequential or chord expression naming all variables in sequence, implying that followup music will be relativized according to the resulting pitch of the last or first variable, respectively.

For constructing the resulting *music* however, the usual copying requirements for avoiding side effects from multiply used music function arguments and return values apply.

An example would be

```
abba =
  #(define-music-function (a b) (ly:music? ly:music?)
    (make-relative (a b)
      #{ #a #b #}
      #{ $a $b $b $a #}))

\relative {
  \abba c'' g'
}
```

`make-repeat name times main alts` [Function]
 Create a repeat music expression, with all properties initialized properly.

`ly:make-rotation` *angle center* [Function]
 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* [Function]
 Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200-cent tones of a pitch above the tonic.

`ly:make-scaling` *scale scaley* [Function]
 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* [Function]
 Encapsulate *music* into a score smob.
 This is a low-level function that does no preprocessing. You might be looking for function `scorify-music` instead, which also preprocesses *music*.

`make-semitone->pitch` *pitches* [Function]
 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 f 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.

`ly:make-skyline` *segments axis direction* [Function]
 Create a new skyline from a list of segments. A skyline is an object representing an outline along a 'horizon axis', much like a city skyline. The argument *segments* is a list of segments. A segment has the form '(x1 . y1) . (x2 . y2)'. The resulting skyline, viewed on the given *axis*, has a building joining these two points for each segment. *x1*, *y1*, *x2*, *y2* may be infinite. The buildings can be given in any order, and overlap.

`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-tmpfile-name` *filename* [Function]

Return *filename*, suffixed by a randomly generated, hexadecimal id.

`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 (concatening them) or on points given either as complex number or real number pair. See also `ly:make-rotation`, `ly:make-scaling`, and `ly:make-translation`.

`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 header or footer markup, with properties reflecting the variables in the page's layout and header modules.

`marked-up-title` *what* [Function]
 Read variable *what* from the page's layout. Interpret it as title markup, with properties reflecting the variable in the page's layout and header modules.

`markup` ... [Macro]
 The markup macro provides a LilyPond-like syntax for building markups using Scheme keywords, replacing `\command` with `#:command`. For example, this:

```

\markup { foo
  \raise #0.2 \hbracket \bold bar
  \override #'(baseline-skip . 4)
  \bracket \column { baz bazr bla }
}

```

translates to this:

```

(markup "foo"
 #:raise 0.2 #:hbracket #:bold "bar"
 #:override '(baseline-skip . 4)
 #:bracket #:column ("baz" "bazr" "bla"))

```

- `markup->string` *m* *#:layout layout* *#:props props* [Function]
 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* [Function]
 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* ... [Function]
 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` ... [Macro]
 Defines and returns an anonymous markup command. Other than not registering the markup command, this is identical to `define-markup-command`.
- `markup-list?` *arg* [Function]
 Return a true value if *x* is a list of markups or markup command lists.
- `markup-list-lambda` ... [Macro]
 Same as `markup-lambda` but defines a markup list command that, when interpreted, returns a list of stencils instead of a single one.
- `matrix-rotate-counterclockwise` *matrix* [Function]
 Return a copy of *matrix* rotated counterclockwise. *matrix* is a 2-dimensional array without non-zero lower bounds in its shape.
- `measure-counter::text` *grob* [Function]
 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* [Function]
 A callback for function `default-flag` to get a mensural flag.
 Mensural flags are aligned with staff lines; for stems ending on staff lines, use different flags than for notes between staff lines. The idea is that the inner end of a flag always touches a staff line.
 The mensural flag glyph is taken from the music font; its name is `flags.mensuralDirTypeLog`. *Dir* is the flag direction (either ‘u’ or ‘d’), *Type* is ‘0’ if the note head is between staff lines and ‘1’ otherwise, *Log* is the duration log (an integer in the range 3 to 6) from which the number of flags attached to the stem is derived. Both *Dir* and *Log* are taken from *grob*. Example: `flags.mensuralu13`.
 This function returns a stencil.
- `ly:message` *str rest* [Function]
 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* [Function]
 Is *spanner* broken and among the middle broken pieces (i.e., neither the first nor the last)?

- `midi-program` *instrument* [Function]
Return the program of the instrument.
- `ly:minimal-breaking` *paper-book* [Function]
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.
- `minmax/cmp` *cmp arg args ...* [Function]
Like `min` or `max`, but applies to any type of values, comparing them with *cmp* instead of `<` or `>`. For example:

```
(minmax/cmp (comparator-from-key string-length <) "a" "aa" "aaa")
⇒ "a"
(minmax/cmp (comparator-from-key string-length >) "a" "aa" "aaa")
⇒ "aaa"
```
- `ly:mm` *num* [Function]
num mm.
- `mmrest-of-length` *mus* [Function]
Create a multi-measure rest of exactly the same length as *mus*.
- `modern-straight-flag` *grob* [Function]
A callback function for `Flag.stencil` to get a modern straight flag.
This is used by composers like Stockhausen or Boulez.
The straight flag angles are 18 and 22 degrees for up-stems and down-stems, respectively, and thus smaller than for `old-straight-flag`. If the caller sets the `stroke-style` property of *grob* to the string "grace", add a slash through the flag.
This function returns a stencil.
- `ly:module->alist` *mod* [Function]
Dump the contents of module *mod* as an alist.
- `ly:module-copy` *dest src* [Function]
Copy all bindings from module *src* into *dest*.
- `ly:modules-lookup` *modules sym def* [Function]
Look up *sym* in the list *modules*, returning the first occurrence. If not found, return *def* or `#f` if *def* isn't specified.
- `ly:moment?` *x* [Function]
Is *x* a smob of class `Moment`?
- `ly:moment<?` *a b* [Function]
Compare two moments.
- `ly:moment-add` *a b* [Function]
Add two moments.
- `ly:moment-div` *a b* [Function]
Divide moment *a* by a number *b* (or by the main part of another moment).
- `ly:moment-grace` *mom* [Function]
Extract grace timing as a rational number from *mom*.
- `ly:moment-grace-denominator` *mom* [Function]
Extract denominator from grace timing.

<code>ly:moment-grace-numerator <i>mom</i></code>	[Function]
Extract numerator from grace timing.	
<code>ly:moment-main <i>mom</i></code>	[Function]
Extract main timing as a rational number from <i>mom</i> .	
<code>ly:moment-main-denominator <i>mom</i></code>	[Function]
Extract denominator from main timing.	
<code>ly:moment-main-numerator <i>mom</i></code>	[Function]
Extract numerator from main timing.	
<code>ly:moment-mod <i>a b</i></code>	[Function]
Modulo of two moments.	
<code>ly:moment-mul <i>a b</i></code>	[Function]
Multiply moment <i>a</i> by a number <i>b</i> (or by the main part of another moment).	
<code>ly:moment-sub <i>a b</i></code>	[Function]
Subtract two moments.	
<code>ly:music? <i>obj</i></code>	[Function]
Is <i>obj</i> a Music object?	
<code>music->make-music <i>obj</i></code>	[Function]
Generate an expression that, once evaluated, may return an object equivalent to <i>obj</i> , that is, for a music expression, a (make-music ...) form.	
<code>music-clone <i>music music-properties</i> ...</code>	[Function]
Clone <i>music</i> and set properties according to <i>music-properties</i> , a list of alternating property symbols and values:	
(music-clone start-span 'span-direction STOP)	
Only properties that are not overridden by <i>music-properties</i> are actually fully cloned.	
<code>ly:music-compress <i>mus scale</i></code>	[Function]
Compress <i>mus</i> by <i>scale</i> .	
<code>ly:music-deep-copy <i>m origin</i></code>	[Function]
Copy <i>m</i> and all sub expressions of <i>m</i> . <i>m</i> may be an arbitrary type; cons cells and music are copied recursively. If <i>origin</i> is given, it is used as the origin for one level of music by calling <code>ly:set-origin!</code> on the copy.	
<code>ly:music-duration-compress <i>mus fact</i></code>	[Function]
Compress <i>mus</i> by factor <i>fact</i> , which is a Moment.	
<code>ly:music-duration-length <i>mus</i></code>	[Function]
Extract the duration field from <i>mus</i> and return the length.	
<code>music-filter <i>pred? music</i></code>	[Function]
Filter out music expressions that do not satisfy <i>pred?</i> .	
<code>ly:music-function? <i>x</i></code>	[Function]
Is <i>x</i> a smob of class Music_function?	
<code>ly:music-function-extract <i>x</i></code>	[Function]
Return the Scheme function inside <i>x</i> .	

<code>ly:music-function-signature x</code>	[Function]
Return the function signature inside <i>x</i> .	
<code>music-is-of-type? mus type</code>	[Function]
Does <i>mus</i> belong to the music class <i>type</i> ?	
<code>ly:music-length mus</code>	[Function]
Get the length of music expression <i>mus</i> and return it as a Moment object.	
<code>ly:music-list? lst</code>	[Function]
Is <i>lst</i> a list of music objects?	
<code>music-map function music</code>	[Function]
Apply <i>function</i> to <i>music</i> and all of the music it contains.	
First it recurses over the children, then the function is applied to <i>music</i> .	
<code>ly:music-mutable-properties mus</code>	[Function]
Return an alist containing the mutable properties of <i>mus</i> . The immutable properties are not available, since they are constant and initialized by the make-music function.	
<code>ly:music-output? x</code>	[Function]
Is <i>x</i> a smob of class Music_output?	
<code>music-pitches music</code>	[Function]
Return a list of all pitches from <i>music</i> .	
<code>ly:music-property mus sym val</code>	[Function]
Return the value for property <i>sym</i> of music expression <i>mus</i> . If no value is found, return <i>val</i> or '() if <i>val</i> is not specified.	
<code>music-selective-filter descend? pred? music</code>	[Function]
Recursively filter out music expressions that do not satisfy <i>pred?</i> , but refrain from filtering the subexpressions of music that does not satisfy <i>descend?</i> .	
<code>music-selective-map descend? function music</code>	[Function]
Apply <i>function</i> recursively to <i>music</i> , but refrain from mapping subexpressions of music that does not satisfy <i>descend?</i> .	
<code>music-separator? m</code>	[Function]
Is <i>m</i> a separator?	
<code>ly:music-set-property! mus sym val</code>	[Function]
Set property <i>sym</i> in music expression <i>mus</i> to <i>val</i> .	
<code>ly:music-start mus</code>	[Function]
Get the start of music expression <i>mus</i> and return it as a Moment object.	
<code>ly:music-transpose m p</code>	[Function]
Transpose <i>m</i> such that central C is mapped to <i>p</i> . Return <i>m</i> .	
<code>music-type-predicate types</code>	[Function]
Return a predicate function that can be used for checking music to have one of the types listed in <i>types</i> .	
<code>neo-modern-accidental-rule context pitch barnum</code>	[Function]
An accidental rule that typesets an accidental if it differs from the key signature <i>and</i> 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` [Function]
 A callback for function `default-flag`, indicating ‘no flag’.
 This function simply returns an 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]
 A callback for function `default-flag` to get a ‘normal’ flag.
 See function `glyph-flag` for the naming scheme of flag glyphs (with argument *flag-style* set to the empty string).
 This function returns a stencil.
- `normalize-color color` [Function]
 Convert a color given in any of the supported formats into a list of 4 numbers: R, G, B, A. Possible formats are: such a list of 4 numbers; a list of 3 numbers (transparency defaults to 1.0); a CSS string (named color, or “#RRGGBB”, or “#RRGGBBAA”, or “#RGB”, or “#RGBA”).
- `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.
 In general, this function can embed arbitrary files into LilyPond’s PDF output (using *embedded file streams*).
- `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 #f, 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.
- `ly:note-scale? x` [Function]
 Is *x* a smob of class Scale?

- `note-to-cluster` *music* [Function]
 Replace NoteEvents by ClusterNoteEvents.
- `ly:number->duration` *x* [Function]
 Convert a duration expressed in units of whole notes to a ly:duration. The log, number of dots, and scaling factor are chosen automatically.
- `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* [Function]
 A callback function for Flag.stencil to get an old straight flag.
 This is used by composers like Bach.
 The up-stem and down-stem angles of the flags are both 45 degrees. If the caller sets the stroke-style property of *grob* to the string "grace", add a slash through the flag.
 This function returns a stencil.
- `ly:one-line-auto-height-breaking` *paper-book* [Function]
 Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.
- `ly:one-line-breaking` *paper-book* [Function]
 Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.
- `ly:one-page-breaking` *paper-book* [Function]
 Put each score on a single page. The paper-height settings are modified so each score fits on one page, and the height of the page matches the height of the full score.
- `ly:optimal-breaking` *paper-book* [Function]
 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* [Function]
 Print ly:set-option usage. Optional *port* argument for the destination defaults to current output port. Specify *internal* to get doc for internal options.
- `optional-material-bracket::positions` *grob* [Function]
 Callback for OptionalMaterialBracket grobs.

- `ly:otf->cff otf-file-name idx` [Function]
 Convert the contents of an OTF file to a CFF file, returning it as a string. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.
- `ly:otf-font? font` [Function]
 Is *font* an OpenType font?
- `ly:otf-font-glyph-info font glyph` [Function]
 Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).
- `ly:otf-font-table-data font tag` [Function]
 Extract a table *tag* from *font*. Return empty string for non-existent *tag*.
- `ly:otf-glyph-count font` [Function]
 Return the number of glyphs in *font*.
- `ly:otf-glyph-list font` [Function]
 Return a list of glyph names for *font*.
- `ly:output-def? x` [Function]
 Is *x* a smob of class `Output_def`?
- `ly:output-def-clone def` [Function]
 Clone output definition *def*.
- `ly:output-def-lookup def sym val` [Function]
 Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `()` if *val* is undefined.
- `ly:output-def-parent output-def default-value` [Function]
 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` [Function]
 Return the variable scope inside *def*.
- `ly:output-def-set-variable! def sym val` [Function]
 Set an output definition *def* variable *sym* to *val*.
- `ly:output-description output-def` [Function]
 Return the description of translators in *output-def*.
- `ly:output-file-name-for-input-file-name name` [Function]
 Convert the *name* of an input file (as provided on the command line) to the name of an output file without an extension, honoring command-line options such as `--output` and `-dstrip-output-dir`.
- `ly:output-find-context-def output-def context-name` [Function]
 Return an alist of all context defs (matching *context-name* if given) in *output-def*.
- `output-module? module` [Function]
 Return `#t` if *module* belongs to an output module usually carrying context definitions (`\midi` or `\layout`).
- `ly:outputter-close outputter` [Function]
 Close port of *outputter*.

<code>ly:outputter-dump-stencil</code> <i>outputter stencil</i>	[Function]
Dump stencil <i>expr</i> onto <i>outputter</i> .	
<code>ly:outputter-dump-string</code> <i>outputter str</i>	[Function]
Dump <i>str</i> onto <i>outputter</i> .	
<code>ly:outputter-output-scheme</code> <i>outputter expr</i>	[Function]
Output <i>expr</i> to the paper outputter.	
<code>ly:outputter-port</code> <i>outputter</i>	[Function]
Return output port for <i>outputter</i> .	
<code>oval-stencil</code> <i>stencil thickness x-padding y-padding</i>	[Function]
Add an oval around <i>stencil</i> , padded by the padding pair, producing a new stencil.	
<code>override-head-style</code> <i>heads style</i>	[Function]
Override style for <i>heads</i> to <i>style</i> .	
<code>override-time-signature-setting</code> <i>time-signature setting</i>	[Function]
Override the time signature settings for the context in <i>time-signature</i> , with the new setting alist <i>setting</i> .	
<code>ly:page-marker?</code> <i>x</i>	[Function]
Is <i>x</i> a smob of class <code>Page_marker</code> ?	
<code>ly:page-turn-breaking</code> <i>paper-book</i>	[Function]
Optimally break (pages and lines) the <code>Paper_book</code> object <i>paper-book</i> such that page turns only happen in specified places, returning its pages.	
<code>ly:pango-font?</code> <i>f</i>	[Function]
Is <i>f</i> a Pango font?	
<code>ly:pango-font-physical-fonts</code> <i>f</i>	[Function]
Return alist of (ps-name file-name font-index) lists for Pango font <i>f</i> .	
<code>pango-pf-file-name</code> <i>pango-pf</i>	[Function]
Return the file name of the Pango physical font <i>pango-pf</i> .	
<code>pango-pf-font-name</code> <i>pango-pf</i>	[Function]
Return the font name of the Pango physical font <i>pango-pf</i> .	
<code>pango-pf-fontindex</code> <i>pango-pf</i>	[Function]
Return the font index of the Pango physical font <i>pango-pf</i> .	
<code>ly:paper-book?</code> <i>x</i>	[Function]
Is <i>x</i> a smob of class <code>Paper_book</code> ?	
<code>ly:paper-book-header</code> <i>pb</i>	[Function]
Return the header definition (<code>\header</code>) in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-pages</code> <i>pb</i>	[Function]
Return pages in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-paper</code> <i>pb</i>	[Function]
Return the paper output definition (<code>\paper</code>) in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-performances</code> <i>pb</i>	[Function]
Return performances in <code>Paper_book</code> object <i>pb</i> .	

- `ly:paper-book-scopes pb` [Function]
Return scopes in `Paper_book` object *pb*.
- `ly:paper-book-systems pb` [Function]
Return systems in `Paper_book` object *pb*.
- `ly:paper-column::break-align-width col align-syms` [Function]
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` [Function]
Optional stencil for `PaperColumn` or `NonMusicalPaperColumn`. Draws the *rank number* of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.
- `ly:paper-fonts def` [Function]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).
- `ly:paper-get-font def chain` [Function]
Find a font metric in output definition *def* satisfying the font qualifiers in alist chain *chain*, and return it. (An alist chain is a list of alists, containing grob properties.)
- `ly:paper-get-number def sym` [Function]
Return the value of variable *sym* in output definition *def* as a double.
- `ly:paper-outputscales def` [Function]
Return the output-scale for output definition *def*.
- `ly:paper-score-paper-systems paper-score` [Function]
Return vector of `paper_system` objects from *paper-score*.
- `ly:paper-system? obj` [Function]
Is *obj* a C++ Prob object of type `paper-system`?
- `parenthesize-stencil stencil half-thickness width angularity padding` [Function]
Add parentheses around *stencil*, returning a new stencil.
- `ly:parse-file name` [Function]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.
- `ly:parse-init name` [Function]
Parse the init file *name*.

- `ly:parse-string-expression` *parser-smob ly-code filename line* [Function]
 Parse the string *ly-code* with *parser-smob*. Return the contained music expression. *filename* and *line* are optional source indicators.
- `parse-terse-string` *terse-definition* [Function]
 Parse a fret-diagram-terse definition string *terse-definition* and return a marking list, which can be used with a fretboard grob.
- `ly:parsed-undead-list!` [Function]
 Return the list of objects that have been found alive but should have been dead, and clear that list.
- `ly:parser-append-to-include-path` *path* [Function]
 Append *path* to the current parser's include path.
- `ly:parser-clear-error` *parser* [Function]
 Clear error flag for *parser*, defaulting to current parser.
- `ly:parser-clone` *closures location* [Function]
 Return a clone of current parser. An association list of port positions to closures can be specified in *closures* in order to have \$ and # interpreted in their original lexical environment. If *location* is a valid location, it becomes the source of all music expressions inside.
- `ly:parser-define!` *symbol val* [Function]
 Bind *symbol* to *val* in current parser's module.
- `ly:parser-error` *msg input* [Function]
 Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.
- `ly:parser-has-error?` *parser* [Function]
 Does *parser* (defaulting to current parser) have an error flag?
- `ly:parser-include-string` *ly-code* [Function]
 Include the string *ly-code* into the input stream for current parser. Can only be used in immediate Scheme expressions (\$ instead of #).
- `ly:parser-lookup` *symbol* [Function]
 Look up *symbol* in current parser's module. Return '() if not defined.
- `ly:parser-output-name` *parser* [Function]
 Return the base name of the output file. If *parser* is left off, use currently active parser.
- `ly:parser-parse-string` *parser-smob ly-code* [Function]
 Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.
- `ly:parser-set-note-names` *names* [Function]
 Replace current note names in parser. *names* is an alist of symbols. This only has effect if the current mode is notes.
- `percussion?` *instrument* [Function]
 Return #t if the instrument should use MIDI channel 9.
- `ly:perform-text-replacements` *props input-string* [Function]
 A string transformer to perform text replacements using the replacement-alist from the property alist chain *props*.

<code>ly:performance-headers</code> <i>performance</i>	[Function]
Return the list of headers with the innermost first.	
<code>ly:performance-write</code> <i>performance filename name</i>	[Function]
Write <i>performance</i> to <i>filename</i> storing <i>name</i> as the name of the performance in the file metadata.	
<code>ly:pitch?</code> <i>x</i>	[Function]
Is <i>x</i> a smob of class Pitch?	
<code>ly:pitch<?</code> <i>p1 p2</i>	[Function]
Is <i>p1</i> lexicographically smaller than <i>p2</i> ?	
<code>ly:pitch-alteration</code> <i>pp</i>	[Function]
Extract the alteration from pitch <i>pp</i> .	
<code>ly:pitch-diff</code> <i>pitch root</i>	[Function]
Return pitch <i>delta</i> such that <i>root</i> transposed by <i>delta</i> equals <i>pitch</i> .	
<code>ly:pitch-negate</code> <i>p</i>	[Function]
Negate pitch <i>p</i> .	
<code>ly:pitch-notename</code> <i>pp</i>	[Function]
Extract the note name from pitch <i>pp</i> .	
<code>ly:pitch-octave</code> <i>pp</i>	[Function]
Extract the octave from pitch <i>pp</i> .	
<code>ly:pitch-quartertones</code> <i>pp</i>	[Function]
Calculate the number of quarter tones of pitch <i>pp</i> from middle C.	
<code>ly:pitch-semitones</code> <i>pp</i>	[Function]
Calculate the number of semitones of pitch <i>pp</i> from middle C.	
<code>ly:pitch-steps</code> <i>p</i>	[Function]
Number of steps counted from middle C of the pitch <i>p</i> .	
<code>ly:pitch-tones</code> <i>pp</i>	[Function]
Calculate the number of tones of pitch <i>pp</i> from middle C as a rational number.	
<code>ly:pitch-transpose</code> <i>p delta</i>	[Function]
Transpose pitch <i>p</i> by the amount <i>delta</i> , where <i>delta</i> is relative to middle C.	
<code>ly:png->eps-dump</code> <i>file-name port r g b a</i>	[Function]
Read the PNG image under <i>file-name</i> and convert it to EPS data, dumping the output onto <i>port</i> . <i>r</i> , <i>g</i> , <i>b</i> and <i>a</i> are the components of the background color.	
<code>ly:png-dimensions</code> <i>file-name</i>	[Function]
Read the PNG image under <i>file-name</i> and return its dimensions as a pair of integers, or <i>#f</i> if there was an error (a warning is printed in this case).	
<code>ly:pointer-group-interface::add-grob</code> <i>grob sym grob-element</i>	[Function]
Add <i>grob-element</i> to <i>grob</i> 's <i>sym</i> grob array.	
<code>polar->rectangular</code> <i>radius angle-in-degrees</i>	[Function]
Return polar coordinates (<i>radius</i> , <i>angle-in-degrees</i>) as rectangular coordinates (x-length . y-length).	

- `ly:position-on-line? sg spos` [Function]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).
- `prepend-alist-chain key val chain` [Function]
Convenience to make a new alist chain from *chain* by prepending a binding of *key* to *val*. This is similar to `acons`, for alist chains (lists of alists).
- `ly:prob? x` [Function]
Is *x* a smob of class Prob?
- `ly:prob-immutable-properties prob` [Function]
Retrieve an alist of immutable properties.
- `ly:prob-mutable-properties prob` [Function]
Retrieve an alist of mutable properties.
- `ly:prob-property prob sym val` [Function]
Return the value for property *sym* of Prob object *prob*. If no value is found, return *val* or '() if *val* is not specified.
- `ly:prob-property? obj sym` [Function]
Is boolean prop *sym* of *obj* set?
- `ly:prob-set-property! obj sym value` [Function]
Set property *sym* of *obj* to *value*.
- `ly:prob-type? obj type` [Function]
Is *obj* the specified prob type?
- `ly:programming-error str rest` [Function]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format`; *rest* holds the formatting arguments (if any).
- `ly:progress str rest` [Function]
A Scheme callable function to print progress *str*. The message is formatted with `format`; *rest* holds the formatting arguments (if any).
- `ly:property-lookup-stats sym` [Function]
Return hash table with a property access corresponding to *sym*. Choices are prob, grob, and context.
- `ly:pt num` [Function]
num printer points.
- `ly:pure-call data grob start end rest` [Function]
Convert property *data* (unpure-pure container or procedure) to value in a pure context defined by *grob*, *start*, *end*, and possibly *rest* arguments.
- `pure-chain-offset-callback grob start end prev-offset` [Function]
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.
- `ratio->fret ratio` [Function]
Calculate a fret number given *ratio* for the harmonic.

`ratio->pitch` *ratio* [Function]
Calculate a pitch given *ratio* for the harmonic.

`read-lily-expression` *chr port* [Function]
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* [Function]
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:regex?` *x* [Function]
Is *x* a smob of class `Regex`?

`ly:regex-exec` *regex string* [Function]
Scan *string* for a match of the regular expression object *regex* (constructed with `ly:make-regex`). Return a match object or `#f`. See `ly:regex-match-...` functions for what you can do with the match object.

For example, this extracts the components of a date in YYYY-MM-DD format:

```
#(define date-components
  (let ((date-regex (ly:make-regex "^((\\d{4})-(\\d{2})-(\\d{2}))$")))
    (lambda (date)
      (let ((match (ly:regex-exec date-regex date)))
        (if match
            (list (string->number (ly:regex-match-substring match 1))
                  (string->number (ly:regex-match-substring match 2))
                  (string->number (ly:regex-match-substring match 3)))
            (error "not a date"))))))
```

`ly:regex-exec->list` *regex string* [Function]
Like `ly:regex-exec`, but return a list of non-overlapping matches instead of the first match only."

`ly:regex-match?` *x* [Function]
Is *x* a regular expression match object?

`ly:regex-match-positions` *match [index]* [Function]
Retrieve the start and end of a capturing group in a regular expression match object, returned as a pair, or `#f`. See `ly:regex-match-substring` for details. The *index* argument is optional, defaulting to 0.

`ly:regex-match-prefix` *m* [Function]
Retrieve the part of the target string before the regex match *m*.

`ly:regex-match-substring` *m [index]* [Function]
Retrieve the substring matched by a specific capturing group in the match object *match*. *index* should be 1 for the first group, 2 for the second group, etc. *index* defaults to 0, which returns the substring matched by the entire regular expression. If the capturing group was not part of the match (e.g., group 2 when matching `aa` against the regex `(a+)|(b+)`), `#f` is returned.

`ly:regex-match-suffix` *m* [Function]
Retrieve the part of the target string after the regex match *m*.

`ly:regex-quote` *string* [Function]

Escape special characters in *string*, forming a regular expression pattern that matches exactly *string*.

Example:

```
(ly:regex-quote "$2")
⇒ "\\$2"
```

`ly:regex-replace` *regex string replacements* [Function]

Scan for matches of the compiled regular expression *regex* (created with `ly:make-regex`) in the string *string*, and form a new string by replacing them according to the *replacements*. Each replacement argument can be:

- A string, which is output as-is.
- A non-negative integer, which is interpreted as a match substring index (see `ly:regex-match-substring`).
- A procedure, which is called on the match object, and should return a string.

This example converts a date from YYYY-MM-DD format to DD-MM-YYYY format:

```
#(define date-yyyy-mm-dd->dd-mm-yyyy
  (let ((date-regex (ly:make-regex "(\\d{4})-(\\d{2})-(\\d{2})")))
    (lambda (date)
      (ly:regex-replace date-regex date 3 "-" 2 "-" 1))))
```

This example does the same, using a procedure:

```
#(define date-yyyy-mm-dd->dd-mm-yyyy
  (let ((date-regex (ly:make-regex "(\\d{4})-(\\d{2})-(\\d{2})")))
    (lambda (date)
      (ly:regex-replace
        date-regex
        date
        (lambda (match)
          (format #f "~a~~a~~a"
                  (ly:regex-match-substring match 3)
                  (ly:regex-match-substring match 2)
                  (ly:regex-match-substring match 1)))))))
```

`ly:regex-split` *regex str* [Function]

Split *str* into non-overlapping occurrences of the regex *regex*, returning a list of the substrings.

`ly:register-stencil-expression` *symbol* [Function]

Add *symbol* as head of a stencil expression.

`ly:register-translator` *creator name description* [Function]

Register a translator *creator* (usually a descriptive alist or a function/closure returning one when given a context argument) with the given symbol *name* and the given *description* alist.

`ly:relative-group-extent` *elements common axis* [Function]

Determine the extent of *elements* relative to *common* in the *axis* direction.

`remove-grace-property` *context-name grob sym* [Function]

Remove all *sym* for *grob* in *context-name*.

`remove-whitespace` *strg* [Function]

Remove characters satisfying `char-whitespace?` from string *strg*.

- `ly:rename-file` *oldname newname* [Function]
 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` [Function]
 Forget all about previously loaded fonts.
- `ly:reset-options` *alist* [Function]
 Reset all program options to the values in *alist*.
- `retrieve-glyph-flag` *flag-style dir dir-modifier grob* [Function]
 Load the correct flag glyph from the music font.
 This is an auxiliary function for `create-glyph-flag`.
- `retrograde-music` *music* [Function]
 Return *music* in retrograde (reversed) order.
- `revert-fontSize` *func-name mag* [Function]
 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* [Function]
 Revert style for *heads*.
- `revert-props` *func-name mag props* [Function]
 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* [Function]
 Make a Stencil object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.
- `ly:round-polygon` *points blot extroversion filled* [Function]
 Make a Stencil object that prints a polygon with corners at the points defined by *points* (a list of coordinate pairs) and roundness *blot*. Optional numeric argument *extroversion* shifts the outline outward, with the default of 0 keeping the middle of the line just on the polygon. If optional Boolean argument *filled* is set to `#t` (which is the default), fill the polygon.
- `rounded-box-stencil` *stencil thickness padding blot* [Function]
 Add a rounded box around *stencil*, producing a new stencil.
- `ly:run-translator` *mus output-def* [Function]
 Process *mus* according to *output-def*. An interpretation context is set up, and *mus* is interpreted with it. The context is returned in its final state.

`scale-beam-thickness` *mag* [Function]

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* [Function]

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* [Function]

Return a clone of *paper*, scaled by the given scale factor.

`scale-props` *func-name mag allowed-to-shrink? props* [Function]

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` [Function]

Is *x* a smob of class `Score`?

`ly:score-add-output-def!` *score def* [Function]

Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Function]

Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout lines.

`ly:score-error? score` [Function]

Was there an error in the score?

`ly:score-header score` [Function]

Return score header.

`ly:score-music score` [Function]

Return score music.

`ly:score-output-defs score` [Function]

All output definitions in a score.

`ly:score-set-header!` *score module* [Function]

Set the score header.

`scorify-music music` [Function]

Preprocess *music* and encapsulate it into a score smob.

Among other things, preprocessing replaces chord repetitions via `'q'` with the correct actual chords.

`ly:script-interface::calc-direction` *grob* [Function]

Usually the 'direction of an articulation is set in default-script-alist. If the 'direction is not specified there and not specified by user settings, this procedure steps in.

In absence of 'direction the direction for vertically placed scripts is calculated taking its 'direction-source, usually Stem, and its 'side-relative-direction, falling back to value 1, into account. For horizontally placed scripts we simply provide LEFT as a fallback value.

As a last resort CENTER is returned.

`ly:script-interface::print` *grob* [Function]

The stencil of a script grob.

`seconds->moment` *s context* [Function]

Return a moment equivalent to *s* seconds at the current tempo.

`select-head-glyph` *style log* [Function]

Select a note head glyph string based on note head style *style* and duration log *log*.

`self-alignment-interface::self-aligned-on-breakable` *grob* [Function]

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 ...* [Function]

Transform sequential music *seq* of type

`<<c d e>>-\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 ...* [Function]

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.

`set-default-paper-size` *name rest ...* [Function]

Set the default paper size to *name* with orientation *rest*.

name is either a predefined paper size string like "quarto" or a pair of numbers like '(cons (* 100 mm) (* 50 mm)) to specify a custom paper size.

If optional argument *rest* is set to 'landscape, pages are rotated by 90 degrees, and wider line widths are set accordingly. Swapping the paper dimensions *without* having the print rotated can be achieved by appending the string landscape to the name of the paper size itself (example: "a4landscape").

This function can only be used at top level; it must come before any \paper block. See also function set-paper-size.

`ly:set-default-scale` *scale* [Function]

Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

`set-global-staff-size` *sz* [Function]

Set the default staff size, where *sz* is thought to be in points.

`ly:set-grob-creation-callback cb` [Function]

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` [Function]

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` [Function]

Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.

`set-mus-properties! m alist` [Function]

Set all of *alist* as properties of *m*.

`ly:set-option var val` [Function]

Set program option *var* to value *val*.

See also function `ly:add-option`.

`ly:set-origin! m origin` [Function]

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` [Function]

Usage example: `\applyoutput #(set-output-property 'Clef 'extra-offset '(0 . 1))`

`set-paper-size name rest ...` [Function]

Set the paper size within `\paper` to *name* with orientation *rest*.

name is either a predefined paper size string like "quarto" or a pair of numbers like `'(cons (* 100 mm) (* 50 mm))` to specify a custom paper size.

If optional argument *rest* is set to 'landscape, pages are rotated by 90 degrees, and wider line widths are set accordingly. Swapping the paper dimensions *without* having the print rotated can be achieved by appending the string landscape to the name of the paper size itself (example: "a4landscape").

This function can only be used within a `\paper` block; it must come before any other functions used within the same `\paper` block. See also function `set-default-paper-size`.

`ly:set-property-cache-callback cb` [Function]

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.

- `ly:set-rand-seed` *seed* [Function]
Seed the internal pseudo-random generator with the specified value.
- `shift-one-duration-log` *music shift dot* [Function]
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* [Function]
Shift an item to the right, but only at the start of the line.
- `shift-semitone->pitch` *key semitone->pitch* [Function]
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* [Function]
Replace *mus* by RestEvent of the same duration if it is a SkipEvent. Useful for extracting parts from crowded scores.
- `skip-of-length` *mus* [Function]
Create a skip of exactly the same length as *mus*.
- `skip-of-moment-span` *start-moment end-moment* [Function]
Make skip music fitting between *start-moment* and *end-moment*. The grace part of *end-moment* matters only if *start-moment* and *end-mom* have the same main part.
- `ly:skyline?` *x* [Function]
Is *x* a smob of class Skyline?
- `ly:skyline->points` *skyline horizon-axis* [Function]
Return a list of points from the given skyline, if viewed with *horizon-axis* as 'horizon axis'. Joining the points with a line draws the outline of the skyline.
- `ly:skyline-distance` *skyline other-skyline horizon-padding* [Function]
Compute the distance between the two skylines, padding by *horizon-padding* if provided.
- `ly:skyline-empty?` *sky* [Function]
Return whether skyline *sky* is empty.
- `ly:skyline-height` *skyline x* [Function]
Return the height of *skyline* at point *x*.
- `ly:skyline-max-height` *skyline* [Function]
Return the maximum height found in *skyline*.
- `ly:skyline-max-height-position` *skyline* [Function]
Return the position at which *skyline* reaches its maximum height.
- `ly:skyline-merge` *skyline1 skyline2* [Function]
Merge the two given skylines.
- `ly:skyline-pad` *skyline horizon-padding* [Function]
Return a version of *skyline* padded by *horizon-padding* along the horizon.
- `ly:skyline-touching-point` *skyline other-skyline horizon-padding* [Function]
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* [Function]
 Return a pair of skylines representing the outline of *stencil*. *axis* is the ‘horizon axis’ (i.e., this function gives skylines suitable for the vertical-skylines property if *axis* is X, and for horizontal-skylines if *axis* is Y).
- `ly:smob-protects` [Function]
 Return LilyPond’s internal smob protection list.
- `ly:solve-spring-rod-problem` *springs rods length ragged* [Function]
 Solve a spring and rod problem for *count* objects that are connected by *count*-1 *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs* argument has the format (ideal, inverse_hook) and *rods* is of the form (idx1, idx2, distance).
length is a number, *ragged* a boolean.
 The function returns a list containing the force (positive for stretching, negative for compressing and #f for non-satisfied constraints) followed by *spring-count*+1 positions of the objects.
- `ly:source-file?` *x* [Function]
 Is *x* a smob of class `Source_file`?
- `ly:source-files` *parser-smob* [Function]
 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* [Function]
 A dummy callback that kills the Grob *grob* if it contains no elements.
- `ly:span-bar::calc-anchor` *grob* [Function]
 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* [Function]
 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* [Function]
 Build the stencil of the span bar.
- `ly:span-bar::print` *grob* [Function]
 The print routine for span bars.
- `ly:span-bar::width` *grob* [Function]
 Compute the width of the SpanBar stencil.
- `Span_stem_engraver` *ctx* [Function]
 Connect cross-staff stems to the stems above in the system.
- `ly:spanner?` *g* [Function]
 Is *g* a spanner object?
- `ly:spanner-bound` *spanner dir def* [Function]
 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* [Function]
 Return broken-into list for *spanner*.

- `ly:spanner-broken-neighbor` *spanner dir* [Function]
Return the broken neighbor of *spanner* on the next or previous system according to *dir*. If there is no neighbor, return #f.
- `ly:spanner-set-bound!` *spanner dir item* [Function]
Set grob *item* as bound in direction *dir* for *spanner*.
- `ly:spawn` *command rest* [Function]
Simple Scheme interface to the GLib function `g_spawn_sync`. If an error occurs, format it with `format` and *rest*.
- `split-list-by-group-lengths` *lst groups* [Function]
Split list into groups whose lengths are given in *groups*. For example:
(`split-list-by-group-lengths` '(a b c d e f) '(3 2 1))
⇒ ((a b c) (d e) (f))
- `split-list-by-separator` *lst pred* [Function]
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* [Function]
Is *x* a smob of class Spring?
- `ly:spring-set-inverse-compress-strength!` *spring strength* [Function]
Set the inverse compress *strength* of *spring*.
- `ly:spring-set-inverse-stretch-strength!` *spring strength* [Function]
Set the inverse stretch *strength* of *spring*.
- `stack-lines` *dir padding baseline stils* [Function]
Stack stencils vertically with a baseline skip.
- `stack-stencil-line` *space stencils* [Function]
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* [Function]
Stack stencils *stils* in direction *axis*, *dir*, using *padding*.
- `stack-stencils-padding-list` *axis dir paddings stils* [Function]
Stack stencils *stils* in direction *axis*, *dir*, using a list of *paddings*.
- `staff-ellipsis::print` *grob* [Function]
Callback for StaffEllipsis grob, which is used with `skipTypesetting`.
- `staff-ellipsis::pure-height` *grob beg end* [Function]
Callback for StaffEllipsis grob, which is used with `skipTypesetting`.
- `ly:staff-symbol-line-thickness` *grob* [Function]
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* [Function]
Return the radius of the staff associated with *grob*.

- `ly:staff-symbol-staff-space grob` [Function]
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` [Function]
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` [Function]
Is *x* a smob of class Stencil?
- `ly:stencil-add args` [Function]
Combine stencils. Takes any number of arguments.
- `ly:stencil-aligned-to stil axis dir` [Function]
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` [Function]
Construct a stencil by putting *second* next to *first*.
axis can be 0 (x axis) or 1 (y axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be '()' or #f. If either of the stencils is spacing (i.e., the result of `\\hspace` markup and similar commands), *padding* does not apply.
- `ly:stencil-empty? stil axis` [Function]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.
- `ly:stencil-expr stil` [Function]
Return the expression of stencil *stil*.
- `ly:stencil-extent stil axis` [Function]
Return a pair of numbers signifying the extent of stencil *stil* in *axis* direction (0 or 1 for x and y axis, respectively).
- `ly:stencil-outline stil outline` [Function]
Return a stencil with the stencil expression (inking) of stencil *stil* but with outline and dimensions from stencil *outline*.
- `stencil-pad-around amount stencil` [Function]
Add a padding of *amount* around *stencil*, returning a new stencil.
- `ly:stencil-rotate stil angle x y` [Function]
Return a stencil *stil* 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 stil angle x y` [Function]
Return a stencil *stil* rotated by *angle* degrees around point (x, y), given in absolute coordinates.
- `ly:stencil-scale stil x y` [Function]
Scale stencil *stil* using the horizontal and vertical scaling factors *x* and optional *y* (defaulting to *x*). Negative values flip or mirror *stil* without changing its origin; this may result in collisions unless it is repositioned.

`ly:stencil-stack` *first axis direction second padding mindist* [Function]

Construct a stencil by stacking *second* next to *first*.

axis can be 0 (x axis) or 1 (y axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be '()' or #f. As opposed to `ly:stencil-combine-at-edge`, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.

If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing (i.e., the result of `\\hspace` markup and similar commands), *padding* and *mindist* do not apply.

`ly:stencil-translate` *stil offset* [Function]

Return a copy of stencil *stil* but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Function]

Return a copy of stencil *stil* but translated by *amount* in *axis* direction.

`stencil-true-extent` *stencil axis* [Function]

Return the extent of the actual printed ink of *stencil* on *axis*.

`stencil-whiteout` *stil* [*style* [*thickness* [*line-thickness* [*color*]]]] [Function]

White-out stencil *stil* (i.e., add a white background around it).

If set, optional argument *style* determines the shape of the white background: value 'outline makes the function use `stencil-whiteout-outline` to produce the white background, value 'rounded-box applies `stencil-whiteout-box` to produce a white background with rounded corners. If other values are given (e.g., 'box) or when unspecified, `stencil-whiteout-box` is used to produce a white background with square corners.

If optional argument *thickness* is specified it determines how far, as a multiple of *line-thickness* (which is an optional argument, too, defaulting to value 0.1), the white background extends past the extents of *stil*. If *thickness* is not specified, an appropriate default is chosen based on *style*.

Optional argument *color* sets the color of the white-out, defaulting to white.

`stencil-whiteout-box` *stil* [*thickness* [*blot* [*color*]]] [Function]

White-out stencil *stil* by printing it on top of a white rectangle.

By default, the white background rectangle encloses *stil* without any borders; this can be changed by setting optional argument *thickness* (defaulting to value zero). Optional argument *blot* gives the blot diameter for rounding the corners of the background rectangle (default value is zero, making sharp corners). Optional argument *color* changes the background color (defaulting to white).

`stencil-whiteout-outline` *stil* [*thickness* [*color* [*angle-increments* [*radial-increments*]]]] [Function]

White-out stencil *stil* by surrounding its outline with white.

This function works by creating a series of white stencils radially offset from the original stencil with angles from 0° to 360°.

Optional argument *thickness* (default value 0.3) specifies how big the white outline is.

The number of angle increments can be controlled with optional argument *angle-increments*, defaulting to value 16 (corresponding to 22.5°). The number of radius increments (from zero to *thickness*) can be controlled with optional argument *radial-increments*, defaulting to value 1. Thus *radial-increments* specifies how many copies of the white stencil we make on our way out to *thickness*, and *angle-increments* how many copies of the white stencil we make between 0° and 360°. In total, *radial-increments***angle-increments* copies are drawn.

Optional argument *color* changes the white-out color (defaulting to white).

- `stencil-with-color` *stencil color* [Function]
 Return a modified version of the given stencil that is colored with the given color. See `normalize-color` for possible color formats.
- `straight-flag` *flag-thickness flag-spacing upflag-angle upflag-length downflag-angle downflag-length* [Function]
 Construct a straight flag stencil function.
 The constructed function expects a single argument, *grob*.
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. If the `stroke-style` property in *grob* is set to the string "grace", add a slash through the flag.
 This is an auxiliary function for `modern-straight-flag`, `old-straight-flag`, and `flat-flag`.
- `ly:stream-event?` *obj* [Function]
 Is *obj* a `Stream_event` object?
- `string->string-list` *str* [Function]
 Convert string *str* into a list of strings with length 1. "aBc" will be converted to ("a" "B" "c"). For an empty string or if *str* is not of type string?, return a list containing "".
- `ly:string-percent-encode` *str* [Function]
 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* [Function]
 Replace string *a* by string *b* in string *s*.
- `style-note-heads` *heads style music* [Function]
 Set *style* for all *heads* in *music*. Works both inside of and outside of chord construct.
- `suggest-convert-ly-message` *version-seen* [Function]
 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 ...* [Function]
 Like `string-concatenate`, but for symbols.
- `ly:system-font-load` *name* [Function]
 Load the OpenType system font *name*.otf. Fonts loaded with this command must contain two additional SFNT font tables called LILC 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.
- `tab-note-head::handle-ties` *grob* [Function]
 Handle tab note heads ending a Tie, deal with RepeatTie.
 If a Slur or Glissando starts at a tie-ending TabNoteHead always print the TabNoteHead parenthesized.
 If a tie-ending TabNoteHead occurs at the beginning of a line print it parenthesized unless sub-properties `note-head-visible` and `parenthesize` of `details.tied-properties` are set #f, which can be done manually or using `hideSplitTiedTabNotes`.

A `TabNoteHead` with `\repeatTie` is printed parenthesized as well, the same holds if `\repeatTie` is applied to a chord. This is useful for *seconda volta* blocks. This behaviour can be switched off with `hideSplitTiedTabNotes`.

- `tab-note-head::print grob` [Function]
 Print a tab note head.
- `tag-group-get tag` [Function]
 Return the tag group (as a list of symbols) that the given *tag* symbol belongs to, `#f` if none.
- `tags-keep-predicate tags` [Function]
 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` [Function]
 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` [Function]
 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` [Function]
 Convert a text markup into a stencil. *layout* is a `\\layout` block. *props* is an alist chain, i.e., a list of alists. *markup* is the markup text to be processed. See also `grob-interpret-markup`.
- `ly:time-signature::print grob` [Function]
 Print routine for time signatures.
- `ly:time-signature::print-x grob` [Function]
 Print routine for an X-shaped sign indicating no time signature.
- `ly:time-tracer-include-and-remove-file file-name` [Function]
 Incorporate records from file *file-name* into the current trace. If successful, remove *file-name*. This supports aggregating completed traces from child processes into the parent's trace.
- `ly:time-tracer-restart name` [Function]
 Reinitialize the global tracer in a child process to avoid interfering with the parent's trace. *name* is the name given to the top-level duration slice in the new trace.
- `ly:time-tracer-set-file file-name` [Function]
 Direct time-trace output to file *file-name*. If *file-name* is `#f`, disable tracing.
- `ly:time-tracer-stop` [Function]
 Finalize the global tracer.
- `to-staff-space size [unit]` [Function]
 Convert absolute *size* in *unit* to staff-space units.
 Possible values for *unit* are 'pt, 'bp, 'mm, 'cm, and 'in. If *unit* is omitted, use 'pt.
 Example:
`\markup \hspace #(to-staff-space 25 'mm)`
- `ly:transform? x` [Function]
 Is *x* a smob of class Transform?
- `ly:transform->list transform` [Function]
 Convert a transform matrix to a list of six values. Values are *xx*, *yx*, *xy*, *yy*, *x0*, *y0*.

- `ly:translate-cpp-warning-scheme str` [Function]
 Translate a string in C++ `printf` format and modify it to use it for Scheme formatting.
- `ly:translator? x` [Function]
 Is *x* a smob of class `Translator`?
- `ly:translator-context trans` [Function]
 Return the context of the translator object *trans*.
- `ly:translator-description creator` [Function]
 Return an alist of properties of translator definition *creator*.
- `ly:translator-group? x` [Function]
 Is *x* a smob of class `Translator_group`?
- `ly:translator-name creator` [Function]
 Return the type name of the translator definition *creator*. The name is a symbol.
- `ly:transpose-key-alist l pit` [Function]
 Make a new key alist of *l* transposed by pitch *pit*.
- `ly:ttf->pfa ttf-file-name idx` [Function]
 Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.
- `ly:ttf-ps-name ttf-file-name idx` [Function]
 Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.
- `ly:tuplet-description? x` [Function]
 Is *x* a smob of class `Tuplet_description`?
- `unbroken-or-first-broken-spanner? spanner` [Function]
 Is *spanner* either unbroken or the first of its broken siblings?
- `unbroken-or-last-broken-spanner? spanner` [Function]
 Is *spanner* either unbroken or the last of its broken siblings?
- `unbroken-spanner? spanner` [Function]
 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` [Function]
 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` [Function]
 Unfold repeats and expand the resulting unfolded-repeated-music.
- `uniq-list lst` [Function]
 Remove doublets from list *lst* (i.e., make its elements unique), assuming that it is sorted. Uses `equal?` for comparisons.

- `uniqued-alist` *alist* [*hash-func* [*assoc-func*]] [Function]
 Make keys unique in *alist*. If duplicate keys are found, the first key-value pair is kept. The order of entries is otherwise preserved. The optional arguments *hash-func* and *assoc-func* are a hashing function and an alist retrieval function, as in Guile's `hashx-...` functions.
- `unity-if-multimeasure` *context dur* [Function]
 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* [Function]
 Convert property *data* (unpure-pure container or procedure) to value in an unpure context defined by *grob* and possibly *rest* arguments.
- `ly:unpure-pure-container?` *x* [Function]
 Is *x* a smob of class `Unpure_pure_container`?
- `ly:unpure-pure-container-pure-part` *pc* [Function]
 Return the pure part of *pc*.
- `ly:unpure-pure-container-unpure-part` *pc* [Function]
 Return the unpure part of *pc*.
- `ly:usage` [Function]
 Print usage message.
- `value-for-spanner-piece` *property args* [Function]
 Associate a piece of broken spanner *grob* with an element of list *arg*.
- `ly:verbose-output?` [Function]
 Was verbose output requested, i.e., is the log level at least `DEBUG`?
- `ly:version` [Function]
 Return the current LilyPond version as a list, e.g., (1 3 127 uu1).
- `ly:version?` *op ver* [Function]
 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` *m* [*id*] [Function]
 Recursively split chords that are separated with `\\`. Optional *id* can be a list of context ids to use. If numeric, they also indicate a voice type override. If *id* is just a single number, that's where numbering starts.
- `volta-bracket::calc-hook-visibility` *bar-glyph* [Function]
 Determine the visibility of the volta bracket end hook, returning #t if *no* hook should be drawn.
- `ly:volta-bracket::calc-shorten-pair` *grob* [Function]
 Calculate the shorten-pair values for an ideal placement of the volta brackets relative to the bar lines.
- `volta-spec-music` *number-list music* [Function]
 Add `\volta` *number-list* to *music*.
- `ly:warning` *str rest* [Function]
 A Scheme callable function to issue the warning *str*. The message is formatted with `format`; *rest* holds the formatting arguments (if any).

- `ly:warning-located` *location str rest* [Function]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with *format*; *rest* holds the formatting arguments (if any).
- `ly:wide-char->utf-8` *wc* [Function]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.
- `write-me` *message* *x* [Function]
Return *x*. Display *message* and write *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	880
add-grace-property	880
add-new-clef	880
add-simple-time-signature-style	881
add-stroke-glyph	881
add-stroke-straight	881
alist->hash-table	881
allow-volta-hook	881
alterations-in-key	881
angle-0-2pi	881
angle-0-360	882
array-copy/subarray!	882
arrow-stencil	882
arrow-stencil-maker	882
assert	882
assoc-get	882
at-bar-line-substitute-caesura-type	882

B

bar-line::calc-break-visibility	883
bar-line::calc-glyph-name	883
bar-line::calc-glyph-name-for-direction	883
bar-line::compound-bar-line	883
bar-line::draw-filled-box	883
bar-line::widen-bar-extent-on-span	883
beam-exceptions	883
beat-base	883
beat-structure	883
bend-spanner::print	884
bend::arrow-head-stencil	883
bend::calc-bend-x-begin	883
bend::calc-bend-x-end	884
bend::target-cautionary	884
bend::text-string	884
bit-list->byte-list	884
bit-list->int	884
book-first-page	884
box-grob-stencil	885
box-stencil	885
bracketify-stencil	885
break-alignable-interface::	
self-alignment-of-anchor	885
break-alignable-interface::	
self-alignment-opposite-of-anchor	885
break-alignment-list	885
byte-list->bit-list	885

C

caesura-script-interface::	
before-line-breaking	885
caesura-to-bar-line-or-divisio	886
caesura-to-divisio	886
calc-harmonic-pitch	886
calc-measure-length	886
centered-spanner-interface::calc-x-offset ..	886
centered-stencil	886
chain-assoc-get	886
change-pitches	886
check-context-path	886
check-grob-path	887
check-music-path	887
chord-name->german-markup	887
chord-name->italian-markup	887
circle-stencil	887
clef-modifier::print	887
clef-transposition-markup	887
collect-book-music-for-book	887
collect-bookpart-for-book	887
collect-music-aux	887
collect-music-for-book	887
comparator-from-key	888
construct-chord-elements	888
context-spec-music	889
copy-repeat-chord	890
count-list	890
create-glyph-flag	890
cross-staff-connect	890
cue-substitute	890
cyclic-base-value	890

D

default-flag	890
define-bar-line	890
define-deprecated-property	891
define-event-class	891
define-event-function	891
define-fonts	891
define-markup-command	891
define-markup-list-command	892
define-music-function	892
define-scheme-function	892
define-syntax-function	892
define-tag-group	892
define-void-function	892
degrees->radians	892
descend-to-context	893

determine-split-list.....	893
determine-string-fret-finger.....	893
dir-basename.....	893
direction-scaled.....	893
display-lily-music.....	893
display-music.....	893
display-scheme-music.....	893
dodecaphonic-no-repeat-rule.....	893
duration-dot-factor.....	894
duration-line::calc.....	894
duration-line::print.....	894
duration-log-factor.....	894
duration-visual.....	894
duration-visual-length.....	894
dynamic-text-spanner::before-line-breaking.....	894

E

elbowed-hairpin.....	895
ellipse-stencil.....	895
end-broken-spanner?.....	895
eval-carefully.....	896
event-chord-notes.....	896
event-chord-pitches.....	896
event-chord-reduce.....	896
event-chord-wrap!.....	896
event-has-articulation?.....	896
expand-repeat-chords!.....	896
expand-repeat-notes!.....	896
extract-beam-exceptions.....	897
extract-music.....	897
extract-named-music.....	897
extract-typed-music.....	897

F

figured-bass-continuation::print.....	897
find-named-props.....	897
find-pitch-entry.....	897
finger-glide::print.....	897
first-assoc.....	897
first-broken-spanner?.....	898
first-member.....	898
flat-flag.....	898
flat-zip-longest.....	898
flatten-list.....	898
flip-stencil.....	898
fold-some-music.....	898
fold-values.....	898
font-name-split.....	899
for-some-music.....	899
format-segno-mark-considering-bar-lines....	899
fret->pitch.....	899
fret-parse-terse-definition-string.....	899
function-chain.....	900

G

generate-crop-stencil.....	900
generate-preview-stencil.....	900
get-bound-note-heads.....	900
get-chord-shape.....	900
get-postscript-bbox.....	900
get-tweakable-music.....	900
glyph-flag.....	901
grob-transformer.....	904
grob::all-objects.....	901
grob::compose-function.....	901
grob::directional-value.....	901
grob::display-objects.....	901
grob::inherit-parent-property.....	901
grob::name.....	901
grob::offset-function.....	901
grob::relay-directional-property.....	902
grob::relay-other-property.....	902
grob::rhythmic-location.....	902
grob::unpure-Y-extent-from-stencil.....	902
grob::when.....	902
group-into-ranges.....	904

H

headers-property-alist-chain.....	905
hook-stencil.....	905
horizontal-script::calc-staff-position.....	905

I

index-map.....	905
int->bit-list.....	906
interpret-markup.....	906
interval-center.....	906
interval-index.....	906
interval-length.....	906
invalidate-alterations.....	906
item::extra-spacing-height-including-staff.....	906

L

layout-line-thickness.....	906
layout-set-absolute-staff-size.....	906
layout-set-staff-size.....	907
left-align-at-split-notes.....	907
lilypond-main.....	907
lilypond-version-outdated?.....	907
list-insert-separator.....	907
list-join.....	907
list-pad-left.....	907
list-pad-right.....	907
lookup-markup-command.....	907
ly:add-context-mod.....	880
ly:add-interface.....	880
ly:add-listener.....	880
ly:add-option.....	880
ly:all-grob-interfaces.....	881
ly:all-options.....	881
ly:all-output-backend-commands.....	881

ly:all-stencil-commands	881	ly:directed	893
ly:all-stencil-expressions	881	ly:disconnect-dispatchers	893
ly:angle	881	ly:dispatcher?	893
ly:append-to-option	882	ly:duration->moment	894
ly:assoc-get	882	ly:duration->number	894
ly:axis-group-interface::add-element	882	ly:duration->string	894
ly:bar-line::calc-anchor	882	ly:duration-compress	894
ly:bar-line::print	883	ly:duration-dot-count	894
ly:base64-encode	883	ly:duration-factor	894
ly:basic-progress	883	ly:duration-log	894
ly:bezier-extent	884	ly:duration-scale	894
ly:bezier-extract	884	ly:duration<?	894
ly:book-add-bookpart!	884	ly:duration?	894
ly:book-add-score!	884	ly:effective-prefix	895
ly:book-book-parts	884	ly:engraver-announce-end-grob	895
ly:book-header	884	ly:engraver-make-grob	895
ly:book-paper	884	ly:engraver-make-item	895
ly:book-process	884	ly:engraver-make-spanner	895
ly:book-process-to-systems	885	ly:engraver-make-sticky	895
ly:book-scores	885	ly:error	896
ly:book-set-header!	885	ly:event-deep-copy	896
ly:book?	884	ly:event-length	896
ly:bp	885	ly:event-property	896
ly:bracket	885	ly:event-set-property!	896
ly:break-alignment-interface::		ly:event?	896
find-nonempty-break-align-group	885	ly:expect-warning	896
ly:broadcast	885	ly:extract-subfont-from-collection	897
ly:cairo-output-stencil	886	ly:find-file	897
ly:cairo-output-stencils	886	ly:font-config-add-directory	898
ly:camel-case->lisp-identifier	886	ly:font-config-add-font	898
ly:chain-assoc-get	886	ly:font-config-display-fonts	898
ly:check-expected-warnings	886	ly:font-config-get-font-file	898
ly:cm	887	ly:font-design-size	898
ly:command-line-code	887	ly:font-file-name	899
ly:command-line-options	887	ly:font-get-glyph	899
ly:connect-dispatchers	888	ly:font-glyph-name-to-index	899
ly:context-children	888	ly:font-index-to-charcode	899
ly:context-current-moment	888	ly:font-magnification	899
ly:context-def-lookup	888	ly:font-metric?	899
ly:context-def-modify	888	ly:font-name	899
ly:context-def?	888	ly:format	899
ly:context-event-source	888	ly:format-output	899
ly:context-events-below	888	ly:generic-bound-extent	900
ly:context-find	888	ly:get-all-function-documentation	900
ly:context-grob-definition	888	ly:get-all-translators	900
ly:context-id	888	ly:get-cff-offset	900
ly:context-matched-pop-property	888	ly:get-context-mods	900
ly:context-mod-apply!	889	ly:get-font-format	900
ly:context-mod?	888	ly:get-option	900
ly:context-name	889	ly:get-spacing-spec	900
ly:context-output-def	889	ly:grob-alist-chain	902
ly:context-parent	889	ly:grob-array->list	902
ly:context-property	889	ly:grob-array-length	902
ly:context-property-pop	889	ly:grob-array-ref	902
ly:context-property-push	889	ly:grob-array?	902
ly:context-property-where-defined	889	ly:grob-basic-properties	902
ly:context-pushpop-property	889	ly:grob-chain-callback	902
ly:context-schedule-moment	889	ly:grob-common-refpoint	902
ly:context-set-property!	889	ly:grob-common-refpoint-of-array	902
ly:context-unset-property	889	ly:grob-default-font	902
ly:context?	888	ly:grob-extent	903
ly:debug	890	ly:grob-get-vertical-axis-group-index	903
ly:default-scale	890	ly:grob-interfaces	903
ly:dimension?	893	ly:grob-layout	903
ly:dir?	893	ly:grob-list->grob-array	903

ly:grob-object.....	903	ly:make-pitch.....	911
ly:grob-original.....	903	ly:make-prob.....	912
ly:grob-parent.....	903	ly:make-rand-seed.....	912
ly:grob-pq<?.....	903	ly:make-regex.....	912
ly:grob-properties?.....	903	ly:make-rotation.....	913
ly:grob-property.....	903	ly:make-scale.....	913
ly:grob-property-data.....	903	ly:make-scaling.....	913
ly:grob-pure-height.....	903	ly:make-score.....	913
ly:grob-pure-property.....	903	ly:make-skyline.....	913
ly:grob-pure-relative-coordinate.....	903	ly:make-spring.....	913
ly:grob-relative-coordinate.....	903	ly:make-stencil.....	913
ly:grob-robust-relative-extent.....	904	ly:make-stream-event.....	913
ly:grob-script-priority-less.....	904	ly:make-tmpfile-name.....	914
ly:grob-set-nested-property!.....	904	ly:make-transform.....	914
ly:grob-set-object!.....	904	ly:make-translation.....	914
ly:grob-set-parent!.....	904	ly:make-unpure-pure-container.....	915
ly:grob-set-property!.....	904	ly:message.....	916
ly:grob-spanned-column-rank-interval.....	904	ly:minimal-breaking.....	917
ly:grob-staff-position.....	904	ly:mm.....	917
ly:grob-suicide!.....	904	ly:module->alist.....	917
ly:grob-system.....	904	ly:module-copy.....	917
ly:grob-translate-axis!.....	904	ly:modules-lookup.....	917
ly:grob-vertical<?.....	904	ly:moment-add.....	917
ly:grob?.....	901	ly:moment-div.....	917
ly:gulp-file.....	904	ly:moment-grace.....	917
ly:gulp-file-utf8.....	904	ly:moment-grace-denominator.....	917
ly:has-glyph-names?.....	905	ly:moment-grace-numerator.....	918
ly:hash-table-keys.....	905	ly:moment-main.....	918
ly:in-event-class?.....	905	ly:moment-main-denominator.....	918
ly:inch.....	905	ly:moment-main-numerator.....	918
ly:input-both-locations.....	905	ly:moment-mod.....	918
ly:input-file-line-char-column.....	905	ly:moment-mul.....	918
ly:input-location?.....	905	ly:moment-sub.....	918
ly:input-message.....	905	ly:moment<?.....	917
ly:input-warning.....	905	ly:moment?.....	917
ly:interpret-music-expression.....	906	ly:music-compress.....	918
ly:intlog2.....	906	ly:music-deep-copy.....	918
ly:item-break-dir.....	906	ly:music-duration-compress.....	918
ly:item-get-column.....	906	ly:music-duration-length.....	918
ly:item?.....	906	ly:music-function-extract.....	918
ly:iterator?.....	906	ly:music-function-signature.....	919
ly:length.....	907	ly:music-function?.....	918
ly:lily-lexer?.....	907	ly:music-length.....	919
ly:lily-parser?.....	907	ly:music-list?.....	919
ly:line-interface::line.....	907	ly:music-mutable-properties.....	919
ly:listened-event-class?.....	907	ly:music-output?.....	919
ly:listened-event-types.....	907	ly:music-property.....	919
ly:listener?.....	907	ly:music-set-property!.....	919
ly:make-book.....	908	ly:music-start.....	919
ly:make-book-part.....	908	ly:music-transpose.....	919
ly:make-context-mod.....	909	ly:music?.....	918
ly:make-dispatcher.....	909	ly:non-fatal-error.....	920
ly:make-duration.....	909	ly:note-column-accidentals.....	920
ly:make-global-context.....	909	ly:note-column-dot-column.....	920
ly:make-global-translator.....	909	ly:note-extra-source-file.....	920
ly:make-grob-properties.....	909	ly:note-head::stem-attachment.....	920
ly:make-listener.....	910	ly:note-scale?.....	920
ly:make-moment.....	910	ly:number->duration.....	921
ly:make-music.....	910	ly:number->string.....	921
ly:make-music-function.....	910	ly:one-line-auto-height-breaking.....	921
ly:make-music-relative!.....	911	ly:one-line-breaking.....	921
ly:make-output-def.....	911	ly:one-page-breaking.....	921
ly:make-page-label-marker.....	911	ly:optimal-breaking.....	921
ly:make-page-permission-marker.....	911	ly:option-usage.....	921
ly:make-paper-putter.....	911	ly:otf->cff.....	922

ly:otf-font-glyph-info.....	922	ly:pitch-transpose.....	926
ly:otf-font-table-data.....	922	ly:pitch<?.....	926
ly:otf-font?.....	922	ly:pitch?.....	926
ly:otf-glyph-count.....	922	ly:png->eps-dump.....	926
ly:otf-glyph-list.....	922	ly:png-dimensions.....	926
ly:output-def-clone.....	922	ly:pointer-group-interface::add-grob.....	926
ly:output-def-lookup.....	922	ly:position-on-line?.....	927
ly:output-def-parent.....	922	ly:prob-immutable-properties.....	927
ly:output-def-scope.....	922	ly:prob-mutable-properties.....	927
ly:output-def-set-variable!.....	922	ly:prob-property.....	927
ly:output-def?.....	922	ly:prob-property?.....	927
ly:output-description.....	922	ly:prob-set-property!.....	927
ly:output-file-name-for-input-file-name....	922	ly:prob-type?.....	927
ly:output-find-context-def.....	922	ly:prob?.....	927
ly:outputter-close.....	922	ly:programming-error.....	927
ly:outputter-dump-stencil.....	923	ly:progress.....	927
ly:outputter-dump-string.....	923	ly:property-lookup-stats.....	927
ly:outputter-output-scheme.....	923	ly:pt.....	927
ly:outputter-port.....	923	ly:pure-call.....	927
ly:page-marker?.....	923	ly:regex-exec.....	928
ly:page-turn-breaking.....	923	ly:regex-exec->list.....	928
ly:pango-font-physical-fonts.....	923	ly:regex-match-positions.....	928
ly:pango-font?.....	923	ly:regex-match-prefix.....	928
ly:paper-book-header.....	923	ly:regex-match-substring.....	928
ly:paper-book-pages.....	923	ly:regex-match-suffix.....	928
ly:paper-book-paper.....	923	ly:regex-match?.....	928
ly:paper-book-performances.....	923	ly:regex-quote.....	929
ly:paper-book-scopes.....	924	ly:regex-replace.....	929
ly:paper-book-systems.....	924	ly:regex-split.....	929
ly:paper-book?.....	923	ly:regex?.....	928
ly:paper-column::break-align-width.....	924	ly:register-stencil-expression.....	929
ly:paper-column::print.....	924	ly:register-translator.....	929
ly:paper-fonts.....	924	ly:relative-group-extent.....	929
ly:paper-get-font.....	924	ly:rename-file.....	930
ly:paper-get-number.....	924	ly:reset-all-fonts.....	930
ly:paper-outputscales.....	924	ly:reset-options.....	930
ly:paper-score-paper-systems.....	924	ly:round-filled-box.....	930
ly:paper-system?.....	924	ly:round-polygon.....	930
ly:parse-file.....	924	ly:run-translator.....	930
ly:parse-init.....	924	ly:score-add-output-def!.....	931
ly:parse-string-expression.....	925	ly:score-embedded-format.....	931
ly:parsed-undead-list!.....	925	ly:score-error?.....	931
ly:parser-append-to-include-path.....	925	ly:score-header.....	931
ly:parser-clear-error.....	925	ly:score-music.....	931
ly:parser-clone.....	925	ly:score-output-defs.....	931
ly:parser-define!.....	925	ly:score-set-header!.....	931
ly:parser-error.....	925	ly:score?.....	931
ly:parser-has-error?.....	925	ly:script-interface::calc-direction.....	932
ly:parser-include-string.....	925	ly:script-interface::print.....	932
ly:parser-lookup.....	925	ly:set-default-scale.....	932
ly:parser-output-name.....	925	ly:set-grob-creation-callback.....	933
ly:parser-parse-string.....	925	ly:set-grob-modification-callback.....	933
ly:parser-set-note-names.....	925	ly:set-middle-C!.....	933
ly:perform-text-replacements.....	925	ly:set-option.....	933
ly:performance-headers.....	926	ly:set-origin!.....	933
ly:performance-write.....	926	ly:set-property-cache-callback.....	933
ly:pitch-alteration.....	926	ly:set-rand-seed.....	934
ly:pitch-diff.....	926	ly:skyline->points.....	934
ly:pitch-negate.....	926	ly:skyline-distance.....	934
ly:pitch-notename.....	926	ly:skyline-empty?.....	934
ly:pitch-octave.....	926	ly:skyline-height.....	934
ly:pitch-quartertines.....	926	ly:skyline-max-height.....	934
ly:pitch-semitones.....	926	ly:skyline-max-height-position.....	934
ly:pitch-steps.....	926	ly:skyline-merge.....	934
ly:pitch-tones.....	926	ly:skyline-pad.....	934

ly:skyline-touching-point	934
ly:skyline?	934
ly:skylines-for-stencil	935
ly:smob-protects	935
ly:solve-spring-rod-problem	935
ly:source-file?	935
ly:source-files	935
ly:span-bar::before-line-breaking	935
ly:span-bar::calc-anchor	935
ly:span-bar::calc-glyph-name	935
ly:span-bar::print	935
ly:span-bar::width	935
ly:spanner-bound	935
ly:spanner-broken-into	935
ly:spanner-broken-neighbor	936
ly:spanner-set-bound!	936
ly:spanner?	935
ly:spawn	936
ly:spring-set-inverse-compress-strength! ..	936
ly:spring-set-inverse-stretch-strength! ..	936
ly:spring?	936
ly:staff-symbol-line-thickness	936
ly:staff-symbol-staff-radius	936
ly:staff-symbol-staff-space	937
ly:stderr-redirect	937
ly:stencil-add	937
ly:stencil-aligned-to	937
ly:stencil-combine-at-edge	937
ly:stencil-empty?	937
ly:stencil-expr	937
ly:stencil-extent	937
ly:stencil-outline	937
ly:stencil-rotate	937
ly:stencil-rotate-absolute	937
ly:stencil-scale	937
ly:stencil-stack	938
ly:stencil-translate	938
ly:stencil-translate-axis	938
ly:stencil?	937
ly:stream-event?	939
ly:string-percent-encode	939
ly:string-substitute	939
ly:system-font-load	939
ly:text-interface::interpret-markup	940
ly:time-signature::print	940
ly:time-signature::print-x	940
ly:time-tracer-include-and-remove-file	940
ly:time-tracer-restart	940
ly:time-tracer-set-file	940
ly:time-tracer-stop	940
ly:transform->list	940
ly:transform?	940
ly:translate-cpp-warning-scheme	941
ly:translator-context	941
ly:translator-description	941
ly:translator-group?	941
ly:translator-name	941
ly:translator?	941
ly:transpose-key-alist	941
ly:ttf->pfa	941
ly:ttf-ps-name	941
ly:tuplet-description?	941
ly:unpure-call	942
ly:unpure-pure-container-pure-part	942
ly:unpure-pure-container-unpure-part	942

ly:unpure-pure-container?	942
ly:usage	942
ly:verbose-output?	942
ly:version	942
ly:version?	942
ly:volta-bracket::calc-shorten-pair	942
ly:warning	942
ly:warning-located	943
ly:wide-char->utf-8	943
lyric-hyphen::vaticana-style	908
lyric-text::print	908

M

make-accidental-dodecaphonic-rule	908
make-accidental-rule	908
make-bow-stencil	908
make-c-time-signature-markup	908
make-circle-stencil	908
make-clef-set	908
make-connected-line	908
make-connected-path-stencil	908
make-cue-clef-set	909
make-cue-clef-unset	909
make-duration-of-length	909
make-ellipse-stencil	909
make-engraver	909
make-filled-box-stencil	909
make-glyph-time-signature-markup	909
make-grob-property-override	909
make-grob-property-revert	910
make-grob-property-set	910
make-harmonic	910
make-line-stencil	910
make-modal-inverter	910
make-modal-transposer	910
make-music	910
make-oval-stencil	911
make-part-combine-context-changes	911
make-part-combine-marks	911
make-partial-ellipse-stencil	911
make-path-stencil	911
make-performer	911
make-relative	912
make-repeat	912
make-semitone->pitch	913
make-stencil-boxer	913
make-stencil-circler	913
make-tmpfile	914
make-translator	914
make-transparent-box-stencil	915
map-selected-alist-keys	915
map-some-music	915
marked-up-headfoot	915
marked-up-title	915
markup	915
markup->string	916
markup-command-list?	916
markup-default-to-string-method	916
markup-lambda	916
markup-list-lambda	916
markup-list?	916
matrix-rotate-counterclockwise	916
measure-counter::text	916

mensural-flag.....	916
middle-broken-spanner?.....	916
midi-program.....	917
minmax/cmp.....	917
mmrest-of-length.....	917
modern-straight-flag.....	917
music->make-music.....	918
music-clone.....	918
music-filter.....	918
music-is-of-type?.....	919
music-map.....	919
music-pitches.....	919
music-selective-filter.....	919
music-selective-map.....	919
music-separator?.....	919
music-type-predicate.....	919

N

neo-modern-accidental-rule.....	919
no-flag.....	920
normal-flag.....	920
normalize-color.....	920
not-first-broken-spanner?.....	920
not-last-broken-spanner?.....	920
note-name->markup.....	920
note-name->string.....	920
note-to-cluster.....	921
number-format.....	921

O

offset-fret.....	921
offsetter.....	921
old-straight-flag.....	921
optional-material-bracket::positions.....	921
output-module?.....	922
oval-stencil.....	923
override-head-style.....	923
override-time-signature-setting.....	923

P

pango-pf-file-name.....	923
pango-pf-font-name.....	923
pango-pf-fontindex.....	923
parenthesize-stencil.....	924
parse-terse-string.....	925
percussion?.....	925
polar->rectangular.....	926
prepend-alist-chain.....	927
pure-chain-offset-callback.....	927

R

ratio->fret.....	927
ratio->pitch.....	928
read-lily-expression.....	928
recording-group-emulate.....	928
remove-grace-property.....	929
remove-whitespace.....	929
retrieve-glyph-flag.....	930
retrograde-music.....	930
revert-fontSize.....	930
revert-head-style.....	930
revert-props.....	930
rounded-box-stencil.....	930

S

scale-beam-thickness.....	931
scale-fontSize.....	931
scale-layout.....	931
scale-props.....	931
scorify-music.....	931
seconds->moment.....	932
select-head-glyph.....	932
self-alignment-interface::	
self-aligned-on-breakable.....	932
sequential-music-to-chord-exceptions.....	932
set-accidental-style.....	932
set-default-paper-size.....	932
set-global-staff-size.....	932
set-mus-properties!.....	933
set-output-property.....	933
set-paper-size.....	933
shift-one-duration-log.....	934
shift-right-at-line-begin.....	934
shift-semitone->pitch.....	934
skip->rest.....	934
skip-of-length.....	934
skip-of-moment-span.....	934
span-bar::compound-bar-line.....	935
Span_stem_engraver.....	935
split-list-by-group-lengths.....	936
split-list-by-separator.....	936
stack-lines.....	936
stack-stencil-line.....	936
stack-stencils.....	936
stack-stencils-padding-list.....	936
staff-ellipsis::print.....	936
staff-ellipsis::pure-height.....	936
stencil-pad-around.....	937
stencil-true-extent.....	938
stencil-whiteout.....	938
stencil-whiteout-box.....	938
stencil-whiteout-outline.....	938
stencil-with-color.....	939
straight-flag.....	939
string->string-list.....	939
style-note-heads.....	939
suggest-convert-ly-message.....	939
symbol-concatenate.....	939

T

tab-note-head::handle-ties	939
tab-note-head::print	940
tag-group-get	940
tags-keep-predicate	940
tags-remove-predicate	940
teaching-accidental-rule	940
to-staff-space	940

U

unbroken-or-first-broken-spanner?	941
unbroken-or-last-broken-spanner?	941
unbroken-spanner?	941
unfold-repeats	941

unfold-repeats-fully	941
uniq-list	941
uniqued-alist	942
unity-if-multimeasure	942

V

value-for-spanner-piece	942
voicify-music	942
volta-bracket::calc-hook-visibility	942
volta-spec-music	942

W

write-me	943
----------------	-----