LilyPond

Notation Reference

The LilyPond development team

This manual provides a reference for all notation that can be produced with LilyPond version 2.21.5. It assumes that the reader is familiar with the material in the Section “Learning Manual” in Learning Manual.

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 http://lilypond.org/.

Copyright © 1998–2020 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. A copy of the license is included in the section entitled “GNU Free Documentation License”.

For LilyPond version 2.21.5
1 Musical notation ............................................. 1

1.1 Pitches ..................................................... 1
  1.1.1 Writing pitches ..................................... 1
    Absolute octave entry .................................. 1
    Relative octave entry .................................. 2
    Accidentals ............................................ 6
    Note names in other languages ....................... 8
  1.1.2 Changing multiple pitches ......................... 10
    Octave checks ......................................... 10
    Transpose .............................................. 11
    Inversion .............................................. 14
    Retrograde ............................................. 14
    Modal transformations ................................. 15
  1.1.3 Displaying pitches ................................. 17
    Clef .................................................. 17
    Key signature ......................................... 22
    Ottava brackets ....................................... 24
    Instrument transpositions ......................... 27
    Automatic accidentals .............................. 29
    Ambitus .............................................. 36
  1.1.4 Note heads .......................................... 38
    Special note heads .................................... 39
    Easy notation note heads ........................... 40
    Shape note heads ..................................... 42
    Improvisation ....................................... 45

1.2 Rhythms .................................................. 46
  1.2.1 Writing rhythms .................................... 46
    Durations .............................................. 47
    Tuplets .............................................. 49
    Scaling durations .................................... 54
    Ties .................................................. 55
  1.2.2 Writing rests ....................................... 59
    Rests ................................................ 59
    Invisible rests ....................................... 61
    Full measure rests ................................... 62
  1.2.3 Displaying rhythms ................................. 66
    Time signature ....................................... 66
    Metronome marks ..................................... 71
    Upbeats .............................................. 74
    Unmetered music ..................................... 75
    Polymetric notation ................................. 77
    Automatic note splitting ............................ 80
    Showing melody rhythms ............................ 82
  1.2.4 Beams ............................................... 84
    Automatic beams ..................................... 84
    Setting automatic beam behavior .................... 87
    Manual beams ....................................... 96
    Feathered beams ................................... 98
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.5 Bars</td>
<td>99</td>
</tr>
<tr>
<td>Bar lines</td>
<td>99</td>
</tr>
<tr>
<td>Bar numbers</td>
<td>107</td>
</tr>
<tr>
<td>Bar and bar number checks</td>
<td>112</td>
</tr>
<tr>
<td>Rehearsal marks</td>
<td>113</td>
</tr>
<tr>
<td>1.2.6 Special rhythmic concerns</td>
<td></td>
</tr>
<tr>
<td>Grace notes</td>
<td></td>
</tr>
<tr>
<td>Aligning to cadenzas</td>
<td></td>
</tr>
<tr>
<td>Time administration</td>
<td></td>
</tr>
<tr>
<td>1.3 Expressive marks</td>
<td></td>
</tr>
<tr>
<td>1.3.1 Expressive marks attached to notes</td>
<td>122</td>
</tr>
<tr>
<td>Articulations and ornamentations</td>
<td></td>
</tr>
<tr>
<td>Dynamics</td>
<td></td>
</tr>
<tr>
<td>New dynamic marks</td>
<td></td>
</tr>
<tr>
<td>1.3.2 Expressive marks as curves</td>
<td>133</td>
</tr>
<tr>
<td>Slurs</td>
<td>133</td>
</tr>
<tr>
<td>Phrasing slurs</td>
<td>136</td>
</tr>
<tr>
<td>Breath marks</td>
<td>138</td>
</tr>
<tr>
<td>Falls and doits</td>
<td>140</td>
</tr>
<tr>
<td>1.3.3 Expressive marks as lines</td>
<td>140</td>
</tr>
<tr>
<td>Glissando</td>
<td>140</td>
</tr>
<tr>
<td>Arpeggio</td>
<td>145</td>
</tr>
<tr>
<td>Trills</td>
<td>148</td>
</tr>
<tr>
<td>1.4 Repeats</td>
<td>151</td>
</tr>
<tr>
<td>1.4.1 Long repeats</td>
<td>151</td>
</tr>
<tr>
<td>Normal repeats</td>
<td>151</td>
</tr>
<tr>
<td>Manual repeat marks</td>
<td>159</td>
</tr>
<tr>
<td>Written-out repeats</td>
<td>161</td>
</tr>
<tr>
<td>1.4.2 Short repeats</td>
<td>163</td>
</tr>
<tr>
<td>Percent repeats</td>
<td>163</td>
</tr>
<tr>
<td>Tremolo repeats</td>
<td>166</td>
</tr>
<tr>
<td>1.5 Simultaneous notes</td>
<td>167</td>
</tr>
<tr>
<td>1.5.1 Single voice</td>
<td>168</td>
</tr>
<tr>
<td>Chorded notes</td>
<td>168</td>
</tr>
<tr>
<td>Chord repetition</td>
<td>170</td>
</tr>
<tr>
<td>Simultaneous expressions</td>
<td>172</td>
</tr>
<tr>
<td>Clusters</td>
<td>173</td>
</tr>
<tr>
<td>1.5.2 Multiple voices</td>
<td>173</td>
</tr>
<tr>
<td>Single-staff polyphony</td>
<td>173</td>
</tr>
<tr>
<td>Voice styles</td>
<td>177</td>
</tr>
<tr>
<td>Collision resolution</td>
<td>178</td>
</tr>
<tr>
<td>Merging rests</td>
<td>183</td>
</tr>
<tr>
<td>Automatic part combining</td>
<td>183</td>
</tr>
<tr>
<td>Writing music in parallel</td>
<td>188</td>
</tr>
<tr>
<td>1.6 Staff notation</td>
<td>191</td>
</tr>
<tr>
<td>1.6.1 Displaying staves</td>
<td>191</td>
</tr>
<tr>
<td>Instantiating new staves</td>
<td>191</td>
</tr>
<tr>
<td>Grouping staves</td>
<td>193</td>
</tr>
<tr>
<td>Nested staff groups</td>
<td>196</td>
</tr>
<tr>
<td>Separating systems</td>
<td>198</td>
</tr>
<tr>
<td>1.6.2 Modifying single staves</td>
<td>199</td>
</tr>
<tr>
<td>Staff symbol</td>
<td>199</td>
</tr>
<tr>
<td>Ossia staves</td>
<td>202</td>
</tr>
<tr>
<td>Hiding staves</td>
<td>205</td>
</tr>
</tbody>
</table>
2 Specialist notation .................................. 274

2.1 Vocal music ........................................ 274
  2.1.1 Common notation for vocal music ..................... 274
  References for vocal music ................................ 274
  Entering lyrics ........................................... 275
  Aligning lyrics to a melody ................................ 276
  Automatic syllable durations .............................. 278
  Manual syllable durations ................................ 280
  Multiple syllables to one note ........................... 282
  Multiple notes to one syllable ............................ 282
  Extenders and hyphens .................................... 286
  Gradual changes of vowel ................................ 286
  2.1.2 Techniques specific to lyrics ...................... 287
  Working with lyrics and variables ....................... 287
  Placing lyrics vertically ................................ 289
  Placing syllables horizontally .......................... 293
  Lyrics and repeats ...................................... 295

1.6.3 Writing parts ........................................ 210
  Instrument names ......................................... 210
  Quoting other voices ..................................... 214
  Formatting cue notes ...................................... 217
  Compressing empty measures ............................. 222

1.7 Editorial annotations .................................. 225
  1.7.1 Inside the staff ................................... 225
  Selecting notation font size ............................. 225
  Fingering instructions ..................................... 229
  Hidden notes .............................................. 231
  Coloring objects ......................................... 232
  Parentheses ................................................ 234
  Stems ....................................................... 234
  1.7.2 Outside the staff .................................. 235
  Note names ................................................ 236
  Balloon help .............................................. 237
  Grid lines ................................................ 238
  Analysis brackets ........................................ 240

1.8 Text ..................................................... 242
  1.8.1 Writing text ......................................... 243
  Text objects overview .................................... 243
  Text scripts ............................................... 245
  Text spanners ............................................. 246
  Text marks ................................................. 248
  Separate text ............................................. 251
  1.8.2 Formatting text ..................................... 252
  Text markup introduction ................................ 252
  Selecting font and font size ............................. 254
  Text alignment ............................................ 257
  Graphic notation inside markup ......................... 262
  Music notation inside markup ........................... 264
  Multi-page markup ...................................... 267
  1.8.3 Fonts ................................................ 267
  Fonts explained ........................................... 267
  Single entry fonts ....................................... 271
  Entire document fonts ................................. 271
Divisi lyrics ................................................................. 303
Polyphony with shared lyrics ................................................. 304
2.2 Keyboard and other multi-staff instruments
2.2.1 Common notation for keyboards ........................................ 339
2.2.2 Piano ................................................................. 346
2.2.3 Accordion ............................................................. 348
2.2.4 Harp ................................................................. 349
2.3 Unfretted string instruments ............................................... 350
2.3.1 Common notation for unfretted strings .............................. 350
2.4 Fretted string instruments .................................................. 353
2.4.1 Common notation for fretted strings .............................. 354
2.1.3 Stanzas ................................................................. 306
Adding stanza numbers .......................................................... 306
Adding dynamics marks to stanzas ........................................... 307
Adding singers’ names to stanzas .............................................. 307
Stanzas with different rhythms ................................................ 308
Printing stanzas at the end ..................................................... 311
Printing stanzas at the end in multiple columns ......................... 312
2.1.4 Songs ................................................................. 313
References for songs ............................................................ 313
Lead sheets ................................................................. 314
2.1.5 Choral ................................................................. 314
References for choral ............................................................ 314
Score layouts for choral .......................................................... 315
2.1.6 Opera and stage musicals ............................................... 317
References for opera and stage musicals ..................................... 318
Character names .............................................................. 318
Musical cues ................................................................. 320
Spoken music ................................................................. 324
Dialogue over music ............................................................ 324
2.1.7 Chants psalms and hymns .............................................. 326
References for chants and psalms .............................................. 326
Setting a chant ............................................................... 326
Pointing a psalm ............................................................. 333
Partial measures in hymn tunes ............................................... 336
2.1.8 Ancient vocal music ................................................... 338
2.2 Keyboard and other multi-staff instruments .......................... 338
2.2.1 Common notation for keyboards ........................................ 339
2.2.2 Piano ................................................................. 346
Piano pedals ................................................................. 347
2.2.3 Accordion ............................................................. 348
Discant symbols .............................................................. 348
2.2.4 Harp ................................................................. 349
References for harps ............................................................ 349
Harp pedals ................................................................. 349
2.3 Unfretted string instruments ............................................... 350
2.3.1 Common notation for unfretted strings .............................. 350
References for unfretted strings .............................................. 351
Bowing indications ............................................................ 351
Harmonics ................................................................. 352
Snap (Bartók) pizzicato ....................................................... 353
2.4 Fretted string instruments .................................................. 353
2.4.1 Common notation for fretted strings .............................. 354
References for fretted strings ................................................. 354
String number indications ..................................................... 354
Default tablatures ............................................................ 355
Custom tablatures ........................................................... 370
Fret diagram markups ........................................................ 373
Predefined fret diagrams ..................................................... 383
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9.2 Ancient notation—common features</td>
<td>448</td>
</tr>
<tr>
<td>Pre-defined contexts</td>
<td>448</td>
</tr>
<tr>
<td>Ligatures</td>
<td>448</td>
</tr>
<tr>
<td>Custodes</td>
<td>449</td>
</tr>
<tr>
<td>2.9.3 Typesetting mensural music</td>
<td>450</td>
</tr>
<tr>
<td>Mensural contexts</td>
<td>450</td>
</tr>
<tr>
<td>Mensural clefs</td>
<td>451</td>
</tr>
<tr>
<td>Mensural time signatures</td>
<td>452</td>
</tr>
<tr>
<td>Mensural note heads</td>
<td>453</td>
</tr>
<tr>
<td>Mensural flags</td>
<td>454</td>
</tr>
<tr>
<td>Mensural rests</td>
<td>454</td>
</tr>
<tr>
<td>Mensural accidentals and key signatures</td>
<td>455</td>
</tr>
<tr>
<td>Annotational accidentals (musica ficta)</td>
<td>455</td>
</tr>
<tr>
<td>White mensural ligatures</td>
<td>456</td>
</tr>
<tr>
<td>2.9.4 Typesetting Gregorian chant</td>
<td>457</td>
</tr>
<tr>
<td>Gregorian chant contexts</td>
<td>457</td>
</tr>
<tr>
<td>Gregorian clefs</td>
<td>458</td>
</tr>
<tr>
<td>Gregorian accidentals and key signatures</td>
<td>459</td>
</tr>
<tr>
<td>Divisiones</td>
<td>459</td>
</tr>
<tr>
<td>Gregorian articulation signs</td>
<td>460</td>
</tr>
<tr>
<td>Augmentum dots (morae)</td>
<td>461</td>
</tr>
<tr>
<td>Gregorian square neume ligatures</td>
<td>461</td>
</tr>
<tr>
<td>2.9.5 Typesetting Kievan square notation</td>
<td>468</td>
</tr>
<tr>
<td>Kievan contexts</td>
<td>468</td>
</tr>
<tr>
<td>Kievan clefs</td>
<td>469</td>
</tr>
<tr>
<td>Kievan notes</td>
<td>469</td>
</tr>
<tr>
<td>Kievan accidentals</td>
<td>470</td>
</tr>
<tr>
<td>Kievan bar line</td>
<td>470</td>
</tr>
<tr>
<td>Kievan melismata</td>
<td>471</td>
</tr>
<tr>
<td>2.9.6 Working with ancient music—scenarios and solutions</td>
<td>472</td>
</tr>
<tr>
<td>Incipits</td>
<td>472</td>
</tr>
<tr>
<td>Mensurstriche layout</td>
<td>473</td>
</tr>
<tr>
<td>Transcribing Gregorian chant</td>
<td>473</td>
</tr>
<tr>
<td>Ancient and modern from one source</td>
<td>476</td>
</tr>
<tr>
<td>Editorial markings</td>
<td>478</td>
</tr>
<tr>
<td>2.10 World music</td>
<td>478</td>
</tr>
<tr>
<td>2.10.1 Common notation for non-Western music</td>
<td>478</td>
</tr>
<tr>
<td>Extending notation and tuning systems</td>
<td>478</td>
</tr>
<tr>
<td>2.10.2 Arabic music</td>
<td>479</td>
</tr>
<tr>
<td>References for Arabic music</td>
<td>479</td>
</tr>
<tr>
<td>Arabic note names</td>
<td>479</td>
</tr>
<tr>
<td>Arabic key signatures</td>
<td>480</td>
</tr>
<tr>
<td>Arabic time signatures</td>
<td>482</td>
</tr>
<tr>
<td>Arabic music example</td>
<td>483</td>
</tr>
<tr>
<td>Further reading for Arabic music</td>
<td>483</td>
</tr>
<tr>
<td>2.10.3 Turkish classical music</td>
<td>484</td>
</tr>
<tr>
<td>References for Turkish classical music</td>
<td>484</td>
</tr>
</tbody>
</table>
3 General input and output ........................................ 486

3.1 Input structure .............................................. 486
  3.1.1 Structure of a score .................................. 486
  3.1.2 Multiple scores in a book .............................. 487
  3.1.3 Multiple output files from one input file .......... 488
  3.1.4 Output file names .................................... 489
  3.1.5 File structure ........................................ 490
3.2 Titles and headers .......................................... 492
  3.2.1 Creating titles headers and footers ............... 492
    Titles explained ........................................ 492
    Default layout of bookpart and score titles .......... 495
    Default layout of headers and footers ................ 498
  3.2.2 Custom titles headers and footers .................. 499
    Custom text formatting for titles ..................... 499
    Custom layout for titles ................................ 500
    Custom layout for headers and footers ............... 503
  3.2.3 Creating output file metadata ....................... 504
  3.2.4 Creating footnotes .................................. 505
    Footnotes in music expressions ....................... 505
    Footnotes in stand-alone text .......................... 510
  3.2.5 Reference to page numbers ............................ 513
  3.2.6 Table of contents ................................... 514
3.3 Working with input files .................................. 517
  3.3.1 Including LilyPond files ............................. 517
    Using variables .......................................... 519
    Using tags ................................................ 520
    Using global settings .................................... 524
  3.3.2 Special characters .............................. 524
    Text encoding ............................................ 525
    Unicode .................................................. 525
    ASCII aliases ............................................. 526
3.4 Controlling output .......................................... 527
  3.4.1 Extracting fragments of music ....................... 527
  3.4.2 Skipping corrected music ............................. 528
  3.4.3 Alternative output formats .......................... 528
    SVG Output ............................................... 529
  3.4.4 Replacing the notation font ....................... 529
3.5 Creating MIDI output ...................................... 531
  3.5.1 Supported notation for MIDI ......................... 531
  3.5.2 Unsupported notation for MIDI ....................... 532
  3.5.3 The MIDI block ...................................... 532
  3.5.4 Controlling MIDI dynamics ............................ 533
    Dynamic marks in MIDI .................................... 533
    Setting MIDI volume ..................................... 534
    Setting MIDI block properties ......................... 537
  3.5.5 Using MIDI instruments ............................... 537
  3.5.6 Using repeats with MIDI .............................. 538
  3.5.7 MIDI channel mapping ................................ 539
  3.5.8 Context properties for MIDI effects ................ 541

Turkish note names ............................................... 484
Turkish key signatures ......................................... 484
Further reading for Turkish music ............................. 485
4 Spacing issues .......................................................... 546

3.5.9 Enhancing MIDI output ............................................. 542
   The \articulate script .............................................. 542
   The \swing script .................................................. 543

3.6 Extracting musical information .................................... 544
3.6.1 Displaying LilyPond notation ................................... 544
3.6.2 Displaying scheme music expressions ......................... 544
3.6.3 Saving music events to a file ................................. 544

4.1 Page layout ......................................................... 546
4.1.1 The \paper block ................................................. 546
4.1.2 Paper size and automatic scaling ............................. 547
   Setting the paper size ............................................ 547
   Automatic scaling to paper size ................................ 548
4.1.3 Fixed vertical spacing \paper variables ....................... 548
4.1.4 Flexible vertical spacing \paper variables .................... 549
   Structure of flexible vertical spacing alists ................... 549
   List of flexible vertical spacing \paper variables ............. 550
4.1.5 Horizontal spacing \paper variables ......................... 551
   \paper variables for widths and margins ..................... 551
   \paper variables for two-sided mode ......................... 552
   \paper variables for shifts and indents ....................... 553
4.1.6 Other \paper variables ....................................... 553
   \paper variables for line breaking ............................ 554
   \paper variables for page breaking ............................ 554
   \paper variables for page numbering ......................... 555
   Miscellaneous \paper variables ................................. 555

4.2 Score layout ...................................................... 556
4.2.1 The \layout block ............................................. 557
4.2.2 Setting the staff size ........................................ 559

4.3 Breaks ............................................................. 560
4.3.1 Line breaking .................................................. 560
4.3.2 Page breaking ................................................ 563
   Manual page breaking ........................................... 563
   Optimal page breaking .......................................... 564
   Minimal page breaking .......................................... 564
   One-page page breaking ....................................... 565
   One-line page breaking ........................................ 565
   One-line-auto-height page breaking ......................... 565
   Optimal page turning ........................................... 565

4.4 Vertical spacing .................................................. 566
4.4.1 Flexible vertical spacing within systems ..................... 567
   Within-system spacing properties ................................ 567
   Spacing of ungrouped staves ................................... 570
   Spacing of grouped staves ...................................... 571
   Spacing of non-staff lines ..................................... 572
4.4.2 Explicit staff and system positioning ......................... 573
4.4.3 Vertical collision avoidance .................................. 581

4.5 Horizontal spacing ............................................... 582
4.5.1 Horizontal spacing overview .................................. 582
4.5.2 New spacing section .......................................... 584
4.5.3 Changing horizontal spacing ................................. 585
   Uniform stretching of tuplets .................................. 586
   Strict note spacing ............................................. 586
4.5.4 Line width .................................................. 587
4.5.5 Proportional notation ....................................... 587
4.6 Fitting music onto fewer pages ............................... 593
4.6.1 Displaying spacing ........................................ 594
4.6.2 Changing spacing .......................................... 595

5 Changing defaults .............................................. 597

5.1 Interpretation contexts ....................................... 597
5.1.1 Contexts explained ........................................ 597
Output definitions - blueprints for contexts .................. 597
Score - the master of all contexts ............................. 598
Top-level contexts - staff containers ......................... 598
Intermediate-level contexts - staves ......................... 598
Bottom-level contexts - voices ............................... 599
5.1.2 Creating and referencing contexts ....................... 599
5.1.3 Keeping contexts alive ................................... 602
5.1.4 Modifying context plug-ins .............................. 605
5.1.5 Changing context default settings ....................... 607
Changing all contexts of the same type ..................... 607
Changing just one specific context ......................... 610
Order of precedence ........................................... 612
5.1.6 Defining new contexts .................................... 612
5.1.7 Context layout order ..................................... 614
5.2 Explaining the Internals Reference ....................... 616
5.2.1 Navigating the program reference ....................... 616
5.2.2 Layout interfaces ........................................ 617
5.2.3 Determining the grob property ......................... 618
5.2.4 Naming conventions ..................................... 619
5.3 Modifying properties ........................................ 619
5.3.1 Overview of modifying properties ....................... 619
5.3.2 The \set command ....................................... 620
5.3.3 The \override command ................................ 621
5.3.4 The \tweak command .................................... 623
5.3.5 \set vs. \override ...................................... 625
5.3.6 The \offset command .................................... 626
5.3.7 Modifying alists ......................................... 631
5.4 Useful concepts and properties ......................... 632
5.4.1 Input modes ............................................... 632
5.4.2 Direction and placement ................................ 634
Articulation direction indicators .......................... 634
The direction property ..................................... 635
5.4.3 Distances and measurements .............................. 635
5.4.4 Dimensions ................................................ 636
5.4.5 Staff symbol properties .................................. 636
5.4.6 Spanners .................................................. 637
Using the \spanner-interface .................................. 637
Using the \line-spanner-interface ............................ 639
5.4.7 Visibility of objects ..................................... 641
Removing the stencil ........................................ 642
Making objects transparent ................................ 642
Painting objects white ....................................... 643
Using break-visibility ....................................... 643
Special considerations ...................................... 645
5.4.8 Line styles ............................................... 648
5.4.9 Rotating objects ................................................................. 648  
Rotating layout objects ......................................................... 649  
Rotating markup ................................................................. 649  
5.5 Advanced tweaks .............................................................. 649  
5.5.1 Aligning objects ............................................................ 650  
Setting X-offset and Y-offset directly ...................................... 650  
Using the side-position-interface .......................................... 651  
Using the self-alignment-interface ......................................... 651  
Using the break-alignable-interface ....................................... 652  
5.5.2 Vertical grouping of grobs .............................................. 654  
5.5.3 Modifying stencils ........................................................ 654  
5.5.4 Modifying shapes ........................................................ 655  
Modifying ties and slurs ....................................................... 655  
5.5.5 Modifying broken spanners ............................................. 659  
Using \alterBroken ............................................................... 659  
5.5.6 Unpure-pure containers ................................................ 661  
5.6 Using music functions ...................................................... 662  
5.6.1 Substitution function syntax ........................................... 663  
5.6.2 Substitution function examples ....................................... 663

Appendix A Notation manual tables ........................................... 666
A.1 Chord name chart ............................................................. 666  
A.2 Common chord modifiers .................................................. 667  
A.3 Predefined string tunings .................................................. 670  
A.4 Predefined fretboard diagrams ........................................... 671  
Diagrams for Guitar ............................................................. 671  
Diagrams for Ukulele ........................................................... 673  
Diagrams for Mandolin ........................................................ 675  
A.5 Predefined paper sizes ..................................................... 677  
A.6 MIDI instruments ............................................................ 680  
A.7 List of colors ................................................................. 681  
A.8 The Emmentaler font ....................................................... 683  
Clef glyphs ........................................................................ 684  
Time Signature glyphs .......................................................... 684  
Number glyphs ................................................................. 685  
Accidental glyphs .............................................................. 685  
Default Notehead glyphs ...................................................... 686  
Special Notehead glyphs ....................................................... 686  
Shape-note Notehead glyphs ................................................ 687  
Rest glyphs ..................................................................... 691  
Flag glyphs ...................................................................... 692  
Dot glyphs ....................................................................... 692  
Dynamic glyphs ............................................................... 693  
Script glyphs ................................................................. 693  
Arrowhead glyphs ............................................................. 696  
Bracket-tip glyphs ............................................................ 696  
Pedal glyphs ................................................................. 696  
Accordion glyphs ............................................................. 696  
Tie glyphs ................................................................... 697  
Vaticana glyphs ............................................................... 697  
Medicea glyphs ............................................................... 698  
Hufnagel glyphs ............................................................... 698  
Mensural glyphs ............................................................. 699  
Neomensural glyphs ......................................................... 702
Appendix B  Cheat sheet ................................. 851
Appendix C  GNU Free Documentation License.............. 854
Appendix D  LilyPond command index......................... 861
Appendix E  LilyPond index ................................. 871
1 Musical notation

This chapter explains how to create musical notation.

1.1 Pitches

This section discusses how to specify the pitch of notes. There are three steps to this process: input, modification, and output.

1.1.1 Writing pitches

This section discusses how to input pitches. There are two different ways to place notes in octaves: absolute and relative mode. In most cases, relative mode will be more convenient.

Absolute octave entry

A pitch name is specified using lowercase letters a through g. The note names c to b are engraved in the octave below middle C.

```latex
{\clef bass
c4 d e f
g4 a b c
d4 e f g}
```

Other octaves may be specified with a single quote (') or comma (,) character. Each ' raises the pitch by one octave; each , lowers the pitch by an octave.

```latex
{
Common octave marks can be entered just once on a reference pitch after \fixed placed before the music. Pitches inside \fixed only need ' or , marks when they are above or below the octave of the reference pitch.

\fixed c' \{ \clef treble
    c'4 e' g' c''
    c'4 g b c'
\}
\clef bass
\fixed c, \{ 
    c,4 e, g, c
    c,4 g,, b,, c,
\}

Pitches in the music expression following \fixed are unaffected by any enclosing \relative, discussed next.

See also
- Snippets: Section “Pitches” in Snippets.

Relative octave entry
Absolute octave entry requires specifying the octave for every single note. Relative octave entry, in contrast, specifies each octave in relation to the last note: changing one note’s octave will affect all of the following notes.

Relative note mode has to be entered explicitly using the \relative command:

\relative startpitch musicexpr

In relative mode, each note is assumed to be as close to the previous note as possible. This means that the octave of each pitch inside musicexpr is calculated as follows:

- If no octave changing mark is used on a pitch, its octave is calculated so that the interval with the previous note is less than a fifth. This interval is determined without considering accidentals.
An octave changing mark ' or , can be added to respectively raise or lower a pitch by an extra octave, relative to the pitch calculated without an octave mark.

Multiple octave changing marks can be used. For example, ' ' and , , will alter the pitch by two octaves.

The pitch of the first note is relative to startpitch. startpitch is specified in absolute octave mode. Which choices are meaningful?

an octave of c

Identifying middle C with c' is quite basic, so finding octaves of c tends to be straightforward. If your music starts with gis above c''', you’d write something like \relative c''' { gis' ... }

an octave of the first note inside

Writing \relative gis''' { gis ... } makes it easy to determine the absolute pitch of the first note inside.

no explicit starting pitch

The form \relative { gis''' ... } serves as a compact version of the previous option: the first note inside is written in absolute pitch itself. (This happens to be equivalent to choosing f as the reference pitch.)

The documentation will usually employ the last option.

Here is the relative mode shown in action:

\relative {
  \clef bass
  c d e f
g a b c
d e f g
}

Octave changing marks are used for intervals greater than a fourth:

\relative {
  c'' g c f,
c' a, e'' c
}

A note sequence without a single octave mark can nevertheless span large intervals:

\relative {
  c f b e
  a d g c
}
When `\relative` blocks are nested, the innermost `\relative` block starts with its own reference pitch independently of the outer `\relative`.

\begin{verbatim}
\relative {
    c' d e f
    \relative {
        c'' d e f
    }
}
\end{verbatim}

`\relative` has no effect on `\chordmode` blocks.

\begin{verbatim}
\new Staff {
    \relative c''' {
        \chordmode { c1 }
    }
    \chordmode { c1 }
}
\end{verbatim}

`\relative` is not allowed inside of `\chordmode` blocks.

Music inside a `\transpose` block is absolute unless a `\relative` is included.

\begin{verbatim}
\relative {
    d' e
    \transpose f g {
        d e
        \relative {
            d' e
        }
    }
}
\end{verbatim}

If the preceding item is a chord, the first note of the chord is used as the reference point for the octave placement of a following note or chord. Inside chords, the next note is always relative to the preceding one. Examine the next example carefully, paying attention to the c notes.

\begin{verbatim}
\relative {
    c'
    <c e g>
}
\end{verbatim}
As explained above, the octave of pitches is calculated only with the note names, regardless of any alterations. Therefore, an E-double-sharp following a B will be placed higher, while an F-double-flat will be placed lower. In other words, a double-augmented fourth is considered a smaller interval than a double-diminished fifth, regardless of the number of semitones that each interval contains.

\relative {
  c''2 fis
  c2 ges
  b2 eisis
  b2 feses
}

In complex situations, it is sometimes useful to get back to a certain pitch regardless of what happened before. This can be done using \resetRelativeOctave:

\resetRelativeOctave c''

See also


Notation Reference: [Octave checks], page 10.

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “RelativeOctaveMusic” in Internals Reference.
Chapter 1: Musical notation

Accidentals

Note: New users are sometimes confused about accidentals and key signatures. In LilyPond, note names specify pitches; key signatures and clefs determine how these pitches are displayed. An unaltered note like c means ‘C natural’, regardless of the key signature or clef. For more information, see Section “Pitches and key signatures” in Learning Manual.

A sharp pitch is made by adding is to the note name, and a flat pitch by adding es. As you might expect, a double sharp or double flat is made by adding isis or eses. This syntax is derived from Dutch note naming conventions. To use other names for accidentals, see [Note names in other languages], page 8.

\relative c'' { ais1 aes aisis aises }

A natural pitch is entered as a simple note name; no suffix is required. A natural sign will be printed when needed to cancel the effect of an earlier accidental or key signature.

\relative c'' { a4 aes a2 }

Quarter tones may be added; the following is a series of Cs with increasing pitches:

\relative c'' { ceseh1 ces ceh c cih cis cisih }

Normally accidentals are printed automatically, but you may also print them manually. A reminder accidental can be forced by adding an exclamation mark ! after the pitch. A cautionary accidental (i.e., an accidental within parentheses) can be obtained by adding the question mark ? after the pitch.

\relative c'' { cis cis cis! cis? c c c! c? }

Accidentals on tied notes are only printed at the beginning of a new system:

\relative c'' {
 cis1~ l~
 \break
 cis
}

/relative c'' { ais1 aes aisis aises }
Selected Snippets

Hiding accidentals on tied notes at the start of a new system

This shows how to hide accidentals on tied notes at the start of a new system.

\relative c' \ { 
    \override Accidental.hide-tied-accidental-after-break = ##t  
    cis1 \ cis \ 
    \break  
    cis 
} 

Preventing extra naturals from being automatically added

In accordance with traditional typesetting rules, a natural sign is printed before a sharp or flat if a previous double sharp or flat on the same note is canceled. To change this behavior to contemporary practice, set the extraNatural property to f in the Staff context.

\relative c' \ { 
    aeses4 aes ais a  
    \set Staff.extraNatural = ##f  
    aeses4 aes ais a 
} 

See also


Notation Reference: [Automatic accidentals], page 29, [Annotational accidentals (musica ficta)], page 455, [Note names in other languages], page 8.

Snippets: Section “Pitches” in Snippets.

**Known issues and warnings**

There are no generally accepted standards for denoting quarter-tone accidentals, so LilyPond’s symbols do not conform to any standard.

**Note names in other languages**

There are predefined sets of note and accidental names for various other languages. Selecting the note name language is usually done at the beginning of the file; the following example is written using Italian note names:

```latex
\language "italiano"

\relative {
    do' re mi sib
}
```

The available languages and the note names they define are:

<table>
<thead>
<tr>
<th>Language</th>
<th>Note Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>c d e f g a bes b</td>
</tr>
<tr>
<td>català or</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>catalan</td>
<td></td>
</tr>
<tr>
<td>deutsch</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>english</td>
<td>c d e f g a bf/b-flat b</td>
</tr>
<tr>
<td>español or</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>espanol</td>
<td></td>
</tr>
<tr>
<td>français</td>
<td>do ré/re mi fa sol la sib si</td>
</tr>
<tr>
<td>italiano</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>norsk</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>português or</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>portugues</td>
<td></td>
</tr>
<tr>
<td>suomi</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>svenska</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>vlaams</td>
<td>do re mi fa sol la sib si</td>
</tr>
</tbody>
</table>

In addition to note names, accidental suffixes may also vary depending on the language:

<table>
<thead>
<tr>
<th>Language</th>
<th>sharp</th>
<th>flat</th>
<th>double sharp</th>
<th>double flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>is</td>
<td>es</td>
<td>isis</td>
<td>eses</td>
</tr>
<tr>
<td>català or</td>
<td>d/s</td>
<td>b</td>
<td>dd/ss</td>
<td>bb</td>
</tr>
<tr>
<td>catalan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deutsch</td>
<td>is</td>
<td>es</td>
<td>isis</td>
<td>eses</td>
</tr>
<tr>
<td>english</td>
<td>s/-sharp</td>
<td>f/-flat</td>
<td>ss/x/-sharpsharp</td>
<td>ff/-flatflat</td>
</tr>
<tr>
<td>español or</td>
<td>s</td>
<td>b</td>
<td>ss/x</td>
<td>bb</td>
</tr>
<tr>
<td>espanol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>français</td>
<td>d</td>
<td>b</td>
<td>dd/x</td>
<td>bb</td>
</tr>
<tr>
<td>italiano</td>
<td>d</td>
<td>b</td>
<td>dd</td>
<td>bb</td>
</tr>
<tr>
<td>norsk</td>
<td>iss/is</td>
<td>ess/es</td>
<td>ississ/isis</td>
<td>essess/eses</td>
</tr>
</tbody>
</table>
In Dutch, German, Norwegian, and Finnish, \texttt{aes} is contracted to \texttt{as}; in Dutch and Norwegian, however, both forms are accepted by LilyPond. Exactly the same holds for \texttt{es} and \texttt{ees}, \texttt{aeses} and \texttt{ases}, and finally \texttt{eeses} and \texttt{eses}.

In German and Finnish, LilyPond additionally provides the more frequent form \texttt{asas} for \texttt{ases}.

\begin{verbatim}
relative c' { a2 as e es a ases e eses }
\end{verbatim}

Some music uses microtones whose alterations are fractions of a ‘normal’ sharp or flat. The following table lists note name suffixes for quarter-tone accidentals; here the prefixes semi- and sesqui- respectively mean ‘half’ and ‘one and a half’.

<table>
<thead>
<tr>
<th>Language</th>
<th>semi-sharp</th>
<th>semi-flat</th>
<th>sesqui-sharp</th>
<th>sesqui-flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>ih</td>
<td>eh</td>
<td>isih</td>
<td>eseh</td>
</tr>
<tr>
<td>català or</td>
<td>qd/qs</td>
<td>qb</td>
<td>tqs/tqqs</td>
<td>tqb</td>
</tr>
<tr>
<td>catalan</td>
<td>ih</td>
<td>eh</td>
<td>isih</td>
<td>eseh</td>
</tr>
<tr>
<td>deutsch</td>
<td>ih</td>
<td>eh</td>
<td>isih</td>
<td>eseh</td>
</tr>
<tr>
<td>english</td>
<td>qs</td>
<td>qf</td>
<td>tqs</td>
<td>tqf</td>
</tr>
<tr>
<td>espanol or</td>
<td>cs</td>
<td>cb</td>
<td>tcs</td>
<td>tcb</td>
</tr>
<tr>
<td>espanol</td>
<td>cs</td>
<td>cb</td>
<td>tcs</td>
<td>tcb</td>
</tr>
<tr>
<td>francais</td>
<td>sd</td>
<td>sb</td>
<td>dsd</td>
<td>bsb</td>
</tr>
<tr>
<td>italiano</td>
<td>sd</td>
<td>sb</td>
<td>dsd</td>
<td>bsb</td>
</tr>
<tr>
<td>norsk</td>
<td>ih</td>
<td>eh</td>
<td>issih/isisih</td>
<td>eseh/eseh</td>
</tr>
<tr>
<td>português or</td>
<td>sqt</td>
<td>bqt</td>
<td>stqt</td>
<td>btqt</td>
</tr>
<tr>
<td>portugues</td>
<td>sqt</td>
<td>bqt</td>
<td>stqt</td>
<td>btqt</td>
</tr>
<tr>
<td>suomi</td>
<td>ih</td>
<td>eh</td>
<td>isih</td>
<td>eseh</td>
</tr>
<tr>
<td>svenska</td>
<td>ih</td>
<td>eh</td>
<td>issih</td>
<td>eseh</td>
</tr>
<tr>
<td>vlaams</td>
<td>hk</td>
<td>hb</td>
<td>khh</td>
<td>bhh</td>
</tr>
</tbody>
</table>

In German, there are similar name contractions for microtones as with normal pitches described above.

\begin{verbatim}
/language "deutsch"
\relativenc' { asah2 eh aih eisih }
\end{verbatim}

Most languages presented here are commonly associated with Western classical music, also referred to as \textit{Common Practice Period}. However, alternate pitches and tuning systems are also supported: see Section 2.10.1 [Common notation for non-Western music], page 478.
1.1.2 Changing multiple pitches

This section discusses how to modify pitches.

Octave checks

In relative mode, it is easy to forget an octave changing mark. Octave checks make such errors easier to find by displaying a warning and correcting the octave if a note is found in an unexpected octave.

To check the octave of a note, specify the absolute octave after the = symbol. This example will generate a warning (and change the pitch) because the second note is the absolute octave d'' instead of d' as indicated by the octave correction.

```
\relative {
  c''2 d=4 d
e2 f
}
```

The octave of notes may also be checked with the \octaveCheck controlpitch command. \octaveCheck is specified in absolute mode. This checks that the interval between the previous note and the controlpitch is within a fourth (i.e., the normal calculation of relative mode). If this check fails, a warning is printed. While the previous note itself is not changed, future notes are relative to the corrected value.

```
\relative {
  c''2 d
  \octaveCheck c'
e2 f
}
```

Compare the two bars below. The first and third \octaveCheck checks fail, but the second one does not fail.

```
\relative {
  c''4 f g f
  c4
  \octaveCheck c'
f
  \octaveCheck c'
}```
See also

Snippets: Section “Pitches” in Snippets.
Internals Reference: Section “RelativeOctaveCheck” in Internals Reference.

Transpose

A music expression can be transposed with $\texttt{\textbackslash transpose}$. The syntax is

\texttt{\textbackslash transpose frompitch topitch musicexpr}

This means that \texttt{musicexpr} is transposed by the interval between the pitches \texttt{frompitch} and \texttt{topitch}: any note with pitch \texttt{frompitch} is changed to \texttt{topitch} and any other note is transposed by the same interval. Both pitches are entered in absolute mode.

\begin{quote}
\textbf{Note:} Music inside a $\texttt{\textbackslash transpose}$ block is absolute unless a $\texttt{\textbackslash relative}$ is included in the block.
\end{quote}

Consider a piece written in the key of D-major. It can be transposed up to E-major; note that the key signature is automatically transposed as well.

\begin{verbatim}
\transpose d e {
  \relative {
    \key d \major
    d'4 fis a d
  }
}
\end{verbatim}

If a part written in C (normal concert pitch) is to be played on the A clarinet (for which an A is notated as a C and thus sounds a minor third lower than notated), the appropriate part will be produced with:

\begin{verbatim}
\transpose a c' {
  \relative {
    \key c \major
    c'4 d e g
  }
}
\end{verbatim}
Note that we specify \texttt{\key c \major} explicitly. If we do not specify a key signature, the notes will be transposed but no key signature will be printed.

\texttt{\transpose} distinguishes between enharmonic pitches: both \texttt{\transpose c cis} or \texttt{\transpose c des} will transpose up a semitone. The first version will print sharps and the notes will remain on the same scale step, the second version will print flats on the scale step above.

\begin{verbatim}
music = \relative { c' d e f }
\new Staff {
   \transpose c cis { \music }
   \transpose c des { \music }
}
\end{verbatim}

\texttt{\transpose} may also be used in a different way, to input written notes for a transposing instrument. The previous examples show how to enter pitches in C (or concert pitch) and typeset them for a transposing instrument, but the opposite is also possible if you for example have a set of instrumental parts and want to print a conductor’s score. For example, when entering music for a B-flat trumpet that begins on a notated E (concert D), one would write:

\begin{verbatim}
musicInBflat = { e4 ... } \transpose c bes, \musicInBflat
\end{verbatim}

To print this music in F (e.g., rearranging to a French horn) you could wrap the existing music with another \texttt{\transpose}:

\begin{verbatim}
musicInBflat = { e4 ... } \transpose f c' { \transpose c bes, \musicInBflat }
\end{verbatim}

For more information about transposing instruments, see [Instrument transpositions], page 27.

**Selected Snippets**

*Transposing pitches with minimum accidentals ("Smart" transpose)*

This example uses some Scheme code to enforce enharmonic modifications for notes in order to have the minimum number of accidentals. In this case, the following rules apply:

- Double accidentals should be removed
- B sharp → C
- E sharp → F
- C flat → B
- F flat → E

In this manner, the most natural enharmonic notes are chosen.

\begin{verbatim}
#(define (naturalize-pitch p)
 (let ((o (ly:pitch-octave p))
   (a (* 4 (ly:pitch-alteration p))))
   ;; alteration, a, in quarter tone steps,
   ;; for historical reasons
   (n (ly:pitch-notename p))))
 (cond
   ((and (> a 1) (or (eqv? n 6) (eqv? n 2)))
    (set! a (- a 2)))

\end{verbatim}
(set! n (+ n 1))
((and (< a -1) (or (eqv? n 0) (eqv? n 3)))
  (set! a (+ a 2))
  (set! n (- n 1)))

(cond
  ((> a 2) (set! a (- a 4)) (set! n (+ n 1)))
  ((< a -2) (set! a (+ a 4)) (set! n (- n 1)))
  (if (< n 0) (begin (set! o (- o 1)) (set! n (+ n 7)))
   (if (> n 6) (begin (set! o (+ o 1)) (set! n (- n 7))))
   (ly:make-pitch o n (/ a 4))))

#(define (naturalize music)
  (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
    (if (pair? es)
        (ly:music-set-property! music 'elements (map naturalize es))
        (if (ly:music? e)
            (ly:music-set-property! music 'element (naturalize e))
            (if (ly:pitch? p)
                (begin
                  (set! p (naturalize-pitch p))
                  (ly:music-set-property! music 'pitch p))
                music))))

naturalizeMusic =
#(define-music-function (m)
  (ly:music? (naturalize m)))

music = \relative c' { c4 d e g }

\score {
  \new Staff {
    \transpose c ais { \music }
    \naturalizeMusic \transpose c ais { \music }
    \transpose c deses { \music }
    \naturalizeMusic \transpose c deses { \music }
  }
  \layout { }
See also
Notation Reference: [Instrument transpositions], page 27, [Inversion], page 14, [Modal transformations], page 15, [Relative octave entry], page 2, [Retrograde], page 14.
Snippets: Section “Pitches” in Snippets.

Known issues and warnings
The relative conversion will not affect \transpose, \chordmode or \relative sections in its argument. To use relative mode within transposed music, an additional \relative must be placed inside \transpose.

Triple accidentals will not be printed if using \transpose. An ‘enharmonically equivalent’ pitch will be used instead (e.g., d-flat rather than e-triple-flat).

Inversion
A music expression can be inverted and transposed in a single operation with:
\inversion around-pitch to-pitch musicexpr

The musicexpr is inverted interval-by-interval around around-pitch, and then transposed so that around-pitch is mapped to to-pitch.

music = \relative { c' d e f }
\new Staff {
\music
\inversion d' d' \music
\inversion d' ees' \music
}

Note: Motifs to be inverted should be expressed in absolute form or be first converted to absolute form by enclosing them in a \relative block.

See also
Notation Reference: [Modal transformations], page 15, [Retrograde], page 14, [Transpose], page 11.

Retrograde
A music expression can be reversed to produce its retrograde:
music = \relative { c'8. ees16( fis8. a16 b8.) gis16 f8. d16 }

\new Staff {
\music
\retrograde \music
}
Known issues and warnings

\texttt{retrograde} is a rather simple tool. Since many events are ‘mirrored’ rather than exchanged, tweaks and directional modifiers for opening spanners need to be added at the matching closing spanners: ` needs to be ended by `), every ` or \texttt{cresc} needs to be ended by `! or \texttt{endcr}, every ` or \texttt{decr} needs to be ended by \texttt{enddecr}. Property-changing commands/overrides with a lasting effect will likely cause surprises.

See also

Notation Reference: [Inversion], page 14, [Modal transformations], page 15, [Transpose], page 11.

Modal transformations

In a musical composition that is based on a scale, a motif is frequently transformed in various ways. It may be transposed to start at different places in the scale or it may be inverted around a pivot point in the scale. It may also be reversed to produce its retrograde, see [Retrograde], page 14.

\begin{example}
\textbf{Note:} Any note that does not lie within the given scale will be left untransformed.
\end{example}

Modal transposition

A motif can be transposed within a given scale with:

\begin{verbatim}
\modalTranspose from-pitch to-pitch scale motif
\end{verbatim}

The notes of \texttt{motif} are shifted within the \texttt{scale} by the number of scale degrees given by the interval between \texttt{to-pitch} and \texttt{from-pitch}:

\begin{verbatim}
diatonicScale = \relative { c' d e f g a b }
motif = \relative { c'8 d e f g a b c }
\end{verbatim}

\begin{verbatim}
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
\end{verbatim}

\begin{music}
\begin{verbatim}
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
\end{verbatim}
\end{music}

An ascending scale of any length and with any intervals may be specified:

\begin{verbatim}
pentatonicScale = \relative { ges aes bes des ees }
motif = \relative { ees'8 des,4 <ges' bes,> <ges bes,> }
\end{verbatim}

\begin{verbatim}
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
\end{verbatim}

\begin{music}
\begin{verbatim}
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
\end{verbatim}
\end{music}
Chapter 1: Musical notation

When used with a chromatic scale \texttt{\textbackslash modalTranspose} has a similar effect to \texttt{\textbackslash transpose}, but with the ability to specify the names of the notes to be used:

\begin{verbatim}
chromaticScale = \relative { c' cis d dis e f fis g gis a ais b }

motif = \relative { c'8 d e f g a b c }
\end{verbatim}

\begin{verbatim}
\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
\end{verbatim}

![Modal inversion](image)

\textbf{Modal inversion}

A motif can be inverted within a given scale around a given pivot note and transposed in a single operation with:

\begin{verbatim}
\modalInversion around-pitch to-pitch scale motif
\end{verbatim}

The notes of \texttt{motif} are placed the same number of scale degrees from the \texttt{around-pitch} note within the \texttt{scale}, but in the opposite direction, and the result is then shifted within the \texttt{scale} by the number of scale degrees given by the interval between \texttt{to-pitch} and \texttt{around-pitch}.

So to simply invert around a note in the scale use the same value for \texttt{around-pitch} and \texttt{to-pitch}:

\begin{verbatim}
octatonicScale = \relative { ees' f fis gis a b c d }

motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }
\end{verbatim}

\begin{verbatim}
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
\end{verbatim}

![Modal inversion](image)

To invert around a pivot between two notes in the scale, invert around one of the notes and then transpose by one scale degree. The two notes specified can be interpreted as bracketing the pivot point:

\begin{verbatim}
scale = \relative { c' g' }

motive = \relative { c' c g' c, }
\end{verbatim}

\begin{verbatim}
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
\end{verbatim}

![Modal inversion](image)
The combined operation of inversion and retrograde produce the retrograde-inversion:
\begin{verbatim}
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }
\end{verbatim}
\new Staff {
\motif
\retrograde \modalInversion c' c' \octatonicScale \motif
}

See also
Notation Reference: [Inversion], page 14, [Retrograde], page 14, [Transpose], page 11.

1.1.3 Displaying pitches
This section discusses how to alter the output of pitches.

Clef
Without any explicit command, the default clef for LilyPond is the treble (or $G$) clef.
\begin{verbatim}
c'2 c'
\end{verbatim}
\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{clef_treble.png}
\caption{Treble clef.}
\end{figure}

However, the clef can be changed by using the \texttt{\clef} command and an appropriate clef name. Middle $C$ is shown in each of the following examples.
\begin{verbatim}
\clef treble
c'2 c'
\clef alto
c'2 c'
\clef tenor
c'2 c'
\clef bass
c'2 c'
\end{verbatim}
\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{clef_examples.png}
\caption{Examples of different clefs.}
\end{figure}

For the full range of possible clef names see Section A.10 [Clef styles], page 706.

Specialized clefs, such as those used in \textit{Ancient} music, are described in [Mensural clefs], page 451, and [Gregorian clefs], page 458. Music that requires tablature clefs is discussed in [Default tablatures], page 355, and [Custom tablatures], page 370.

For mixing clefs when using cue notes, see the \texttt{\cujęClef} and \texttt{\cujęDuringWithClef} commands in [Formatting cue notes], page 217.
By adding _8 or ^8 to the clef name, the clef is transposed one octave down or up respectively, and _15 and ^15 transpose by two octaves. Other integers can be used if required. Clef names containing non-alphabetic characters must be enclosed in quotes.

\clef treble
c'2 c'
\clef "treble_8"
c'2 c'
\clef "bass^15"
c'2 c'
\clef "alto_2"
c'2 c'
\clef "G_8"
c'2 c'
\clef "F^5"
c'2 c'

Optional octavation can be obtained by enclosing the numeric argument in parentheses or brackets:

\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'

The pitches are displayed as if the numeric argument were given without parentheses/brackets.

By default, a clef change taking place at a line break will cause the new clef symbol to be printed at the end of the previous line, as a warning clef, as well as the beginning of the next. This warning clef can be suppressed.

\clef treble { c'2 c' } \break
\clef bass { c'2 c' } \break
\clef alto
\set Staff.explicitClefVisibility = #end-of-line-invisible
{ c'2 c' } \break
\unset Staff.explicitClefVisibility
\clef bass { c'2 c' } \break
By default, a clef that has previously been printed will not be re-printed if the same \clef command is issued again and will be ignored. The command $\set \text{Staff.forceClef} = ##t$ changes this behaviour.

\clef treble
c'1
\clef treble
c'1
\set \text{Staff.forceClef} = ##t
c'1
\clef treble
c'1

To be more precise, it is not the \clef command itself that prints a clef. Instead, it sets or changes a property of the Clef_engraver, which then decides by its own whether to display a clef or not in the current staff. The forceClef property overrides this decision locally to re-print a clef once.

When there is a manual clef change, the glyph of the changed clef will be smaller than normal. This behaviour can be overridden.

\clef "treble"
c'1
\clef "bass"
c'1
\clef "treble"
c'1
\override \text{Staff.Clef.full-size-change} = ##t
\clef "bass"
c'1
\clef "treble"
c'1
\revert \text{Staff.Clef.full-size-change}
\clef "bass"
c'1
\clef "treble"
c'1
Selected Snippets

Tweaking clef properties

Changing the Clef glyph, its position, or the ottavation does not change the position of subsequent notes on the staff. To get key signatures on their correct staff lines `middleCClefPosition` must also be specified, with positive or negative values moving middle C up or down respectively, relative to the staff’s center line.

For example, `{clef "treble_8"` is equivalent to setting the `clefGlyph`, `clefPosition` (the vertical position of the clef itself on the staff), `middleCPosition` and `clefTransposition`. Note that when any of these properties (except `middleCPosition`) are changed a new clef symbol is printed.

The following examples show the possibilities when setting these properties manually. On the first line, the manual changes preserve the standard relative positioning of clefs and notes, whereas on the second line, they do not.

```plaintext
{% The default treble clef
\key f \major
c'1
%
% The standard bass clef
\set Staff.clefGlyph = "#clefs.F"
\set Staff.clefPosition = #2
\set Staff.middleCPosition = #6
\set Staff.middleCClefPosition = #6
\key g \major
c'1
%
% The baritone clef
\set Staff.clefGlyph = "#clefs.C"
\set Staff.clefPosition = #4
\set Staff.middleCPosition = #4
\set Staff.middleCClefPosition = #4
\key f \major
c'1
%
% The standard choral tenor clef
\set Staff.clefGlyph = "#clefs.G"
\set Staff.clefPosition = #-2
\set Staff.clefTransposition = #-7
\set Staff.middleCPosition = #1
\set Staff.middleCClefPosition = #1
\key f \major
c'1
%
% A non-standard clef
\set Staff.clefPosition = #0
\set Staff.clefTransposition = #0
\set Staff.middleCPosition = #0
\set Staff.middleCClefPosition = #0
\key g \major
c'1 \break
%
% The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs:
```
\set Staff.clefGlyph = "#clefs.F"
\set Staff.clefPosition = #2
c'1
\set Staff.clefGlyph = "#clefs.G"
c'1
\set Staff.clefGlyph = "#clefs.C"
c'1
\set Staff.clefTransposition = #7
c'1
\set Staff.clefTransposition = #0
\set Staff.clefPosition = #0
c'1

% Return to the normal clef:
\set Staff.middleCPosition = #0
c'1

\new Staff \with {
\override Clef.color = #blue
\override ClefModifier.color = #red
}
\clef "treble_8" c'4

See also

Notation Reference: [Mensural clefs], page 451, [Gregorian clefs], page 458, [Default tablatures], page 355, [Custom tablatures], page 370, [Formatting cue notes], page 217.

Installed Files: scm/parser-clef.scm.

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “Clef_engraver” in Internals Reference, Section “Clef” in Internals Reference, Section “ClefModifier” in Internals Reference, Section “clef-interface” in Internals Reference.

Known issues and warnings

Ottavation numbers attached to clefs are treated as separate grobs. So any \override done to the Clef will also need to be applied, as a separate \override, to the ClefModifier grob.

\new Staff \with {
\override Clef.color = #blue
\override ClefModifier.color = #red
}
\clef "treble_8" c'4
Key signature

Note: New users are sometimes confused about accidentals and key signatures. In LilyPond, note names are the raw input; key signatures and clefs determine how this raw input is displayed. An unaltered note like c means ‘C natural’, regardless of the key signature or clef. For more information, see Section “Pitches and key signatures” in Learning Manual.

The key signature indicates the tonality in which a piece is played. It is denoted by a set of alterations (flats or sharps) at the start of the staff. The key signature may be altered:

\key pitch mode

Here, mode should be \major or \minor to get a key signature of pitch-major or pitch-minor, respectively. You may also use the standard mode names, also called church modes: \ionian, \dorian, \phrygian, \lydian, \mixolydian, \aeolian, and \locrian.

\relative {
  \key g \major
  fis'1
  f
  fis
}

Additional modes can be defined, by listing the alterations for each scale step when the mode starts on C.

freygish = #^((0 . ,NATURAL) (1 . ,FLAT) (2 . ,NATURAL)
(3 . ,NATURAL) (4 . ,NATURAL) (5 . ,FLAT) (6 . ,FLAT))

\relative {
  \key c \freygish c'4 des e f
  \bar "||" \key d \freygish d es fis g
}

Accidentals in the key signature may be printed in octaves other than their traditional positions, or in multiple octaves, by using the flat-positions and sharp-positions properties of KeySignature. Entries in these properties specify the range of staff-positions where accidentals will be printed. If a single position is specified in an entry, the accidentals are placed within the octave ending at that staff position.

\override Staff.KeySignature.flat-positions = #'((-5 . 5))
\override Staff.KeyCancellation.flat-positions = #'((-5 . 5))
\clef bass \key es \major es g bes d'
\clef treble \bar "||" \key es \major es' g' bes' d''

\override Staff.KeySignature.sharp-positions = #'(2)
Chapter 1: Musical notation

Selected Snippets

*Preventing natural signs from being printed when the key signature changes*

When the key signature changes, natural signs are automatically printed to cancel any accidentals from previous key signatures. This may be prevented by setting to `f` the `printKeyCancellation` property in the `Staff` context.

\begin{verbatim}
\relative c' {
    \key d \major
    a4 b cis d
    \key g \minor
    a4 bes c d
    \set Staff.printKeyCancellation = ##f
    \key d \major
    a4 b cis d
    \key g \minor
    a4 bes c d
}
\end{verbatim}

*Non-traditional key signatures*

The commonly used `\key` command sets the `keyAlterations` property in the `Staff` context. To create non-standard key signatures, set this property directly.

The format of this command is a list:

\begin{verbatim}
\set Staff.keyAlterations = #`(((octave . step) . alter) ((octave . step) . alter) ...)
\end{verbatim}

where, for each element in the list `octave` specifies the octave (0 being the octave from middle c to the b above), `step` specifies the note within the octave (0 means c and 6 means b), and `alter` is `SHARP`, `FLAT`, `DOUBLE-SHARP` etc.

Alternatively, using the more concise format for each item in the list, `(step . alter)` specifies the same alteration holds in all octaves. For microtonal scales where a “sharp” is not 100 cents, `alter` refers to the proportion of a 200-cent whole tone.

\begin{verbatim}
\include "arabic.ly"
\relative do' {
    \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT) (1 . ,SEMI-FLAT) (2 . ,FLAT) (5 . ,FLAT) (6 . ,SEMI-FLAT))

    %\set Staff.extraNatural = ##f
    re reb \dwn reb resd
dod dob dosd \dwn dob |
\end{verbatim}
Ottava brackets introduce an extra transposition of an octave for the staff:

\relative c'' { 
  a2 b 
  \ottava #-2 a2 b 
  \ottava #-1 a2 b 
  \ottava #0 a2 b 
  \ottava #1 a2 b 
  \ottava #2 a2 b 
}

By default, only a number is printed at the start of the bracket. That setting may be changed to include an abbreviated ordinal, either in superscript or in normal letters (the latter was formerly used by default); the default bold font weight of these characters may also be altered, as explained in [Selecting font and font size], page 254.

The following example demonstrates various options, as well as how to go back to the current default behavior:

\relative c'' { 
  \ottava #1 
  a'2 b
Selecting Snippets

Changing ottava text

Internally, `\ottava` sets the properties `ottavation` (for example, to 8va or 8vb) and `middleCPosition`. To override the text of the bracket, set `ottavation` after invoking `\ottava`.

Short text is especially useful when a brief ottava is used.

```
{  
c'2  
\ottava #1  
\set Staff.ottavation = #'8'  
c'2  
\ottava #0  
c'1  
\ottava #1  
\set Staff.ottavation = #'Text'  
c'1  
}
```
**Adding an ottava marking to a single voice**

If you have more than one voice on the staff, setting octavation in one voice will transpose the position of notes in all voices for the duration of the ottava bracket. If the ottavation is only intended to apply to one voice, the middleCPosition and ottava bracket may be set explicitly. In this snippet, the bass clef usually has middleCPosition set to 6, six positions above the center line, so in the 8va portion middleCPosition is 7 positions (one octave) higher still.

```
{  
clef bass
<< { <g d'>1~ q2 <c' e'> } 
\r2.
\set Staff.ottavation = #"8vb"
\once \override Staff.OttavaBracket.direction = #DOWN
\set Voice.middleCPosition = #(+ 6 7)
<b,,, b,,>4 ~ |
q2 
\unset Staff.ottavation
\unset Voice.middleCPosition
\<c e>2
}
} >>
```

*Modifying the Ottava spanner slope*

It is possible to change the slope of the Ottava spanner.

```
\relative c'' {  
\override Staff.OttavaBracket.stencil = #ly:line-spanner::print
\override Staff.OttavaBracket.bound-details = #"((left . ((Y . 0) ; Change the integer here
(attach-dir . ,LEFT)
(padding . 0)
(stencil-align-dir-y . ,CENTER)))
(right . ((Y . 5) ; Change the integer here
(attach-dir . ,RIGHT)
(text . ,(make-draw-dashed-line-markup (cons 0 -1.2))))))
\override Staff.OttavaBracket.left-bound-info = 
#ly:line-spanner::calc-left-bound-info-and-text
\override Staff.OttavaBracket.right-bound-info = 
#ly:line-spanner::calc-right-bound-info
\ottava #1
\c1
\c''''1
```
When typesetting scores that involve transposing instruments, some parts can be typeset in a different pitch than the concert pitch. In these cases, the key of the transposing instrument should be specified; otherwise the MIDI output and cues in other parts will produce incorrect pitches. For more information about quotations, see [Quoting other voices], page 214.

\transposition

The pitch to use for \transposition should correspond to the real sound heard when a c’ written on the staff is played by the transposing instrument. This pitch is entered in absolute mode, so an instrument that produces a real sound which is one tone higher than the printed music should use \transposition d’. \transposition should only be used if the pitches are not being entered in concert pitch.

Here are a few notes for violin and B-flat clarinet where the parts have been entered using the notes and key as they appear in each part of the conductor’s score. The two instruments are playing in unison.

\new GrandStaff <<
\new Staff = "violin" \with {
  instrumentName = "Vln"
  midiInstrument = "violin"
}
\relative c'' { % not strictly necessary, but a good reminder
  \transposition c'
  \key c \major
  g4( c8) r c r c4
}\new Staff = "clarinet" \with {
  instrumentName = \markup { Cl (B\flat) }
  midiInstrument = "clarinet"
}
\relative c'' {
  \transposition bes
  \key d \major
  a4( d8) r d r d4
}
The \texttt{\textbackslash transposition} may be changed during a piece. For example, a clarinetist may be required to switch from an A clarinet to a B-flat clarinet.

\begin{verbatim}
flute = \relative c'' { \\
  \key f \major \\
  \cueduring "clarinet" \down { \\
    R1 _\markup \tiny "clarinet" \\
    c4 f e d \\
    R1 _\markup \tiny "clarinet" 
  }
}

clarinet = \relative c'' { \\
  \key aes \major \\
  \transposition a \\
  aes4 bes c des \\
  R1^\markup \{ muta in B\flat \} \\
  \key g \major \\
  \transposition bes \\
  d2 g,
}
\addquote "clarinet" \clarinet
\end{verbatim}

See also


Notation Reference: [Quoting other voices], page 214, [Transpose], page 11.

Snippets: Section “Pitches” in \textit{Snippets}.
Automatic accidentals

There are many different conventions on how to typeset accidentals. LilyPond provides a function to specify which accidental style to use. This function is called as follows:

\new Staff <<
  \accidentalStyle voice
  { ... }
>>

The accidental style applies to the current Staff by default (with the exception of the styles piano and piano-cautionary, which are explained below). Optionally, the function can take a second argument that determines in which scope the style should be changed. For example, to use the same style in all staves of the current StaffGroup, use:

\accidentalStyle StaffGroup.voice

The following accidental styles are supported. To demonstrate each style, we use the following example:

musicA = {
  <<
    \relative {
      cis''8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \
    \relative {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative {
      <fis a cis>8[ <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
      <fis, a> <fis a>]
      \showStaffSwitch
      \change Staff = up
      dis'4 |
      \change Staff = down
      <fis, a cis>4 gis <f a d>2 |
    }
  }
}

\new PianoStaff {
  <<
    \new Staff = "up" {
      \accidentalStyle default
Note that the last lines of this example can be replaced by the following, as long as the same accidental style should be used in both staves.

\new PianoStaff {
  \new Staff = "up" {
    \\ accidentalStyle Score.default
    \musicA
  }
  \\ accidentalStyle Score.default
  \\ change the next line as desired:
  \\ accidentalStyle Score.default
  \\ \musicA
  \new Staff = "down" {
    \musicB
  }
  \new Staff = "down" {
    \musicB
  }
  \\}
}

default

This is the default typesetting behavior. It corresponds to eighteenth-century common practice: accidentals are remembered to the end of the measure in which they occur and only in their own octave. Thus, in the example below, no natural signs are printed before the b in the second measure or the last c:

\new PianoStaff {
  \\ change the next line as desired:
  \\ accidentalStyle Score.default
  \\ accidentalStyle Score.default
  \\ \musicB
  \new Staff = "up" {
    \musicA
  }
  \new Staff = "down" {
    \musicB
  }
  \\}
}

voice

The normal behavior is to remember the accidentals at Staff-level. In this style, however, accidentals are typeset individually for each voice. Apart from that, the rule is similar to default.

As a result, accidentals from one voice do not get canceled in other voices, which is often an unwanted result: in the following example, it is hard to determine whether the second a should be played natural or sharp. The voice option should therefore be used only if the voices are to be read solely by individual musicians. If the staff
is to be used by one musician (e.g., a conductor or in a piano score) then modern or modern-cautionary should be used instead.

modern

This rule corresponds to the common practice in the twentieth century. It omits some extra natural signs, which were traditionally prefixed to a sharp following a double sharp, or a flat following a double flat. The modern rule prints the same accidentals as default, with two additions that serve to avoid ambiguity: after temporary accidentals, cancellation marks are printed also in the following measure (for notes in the same octave) and, in the same measure, for notes in other octaves. Hence the naturals before the \( b \) and the \( c \) in the second measure of the upper staff:

modern-cautionary

This rule is similar to modern, but the ‘extra’ accidentals are printed as cautionary accidentals (with parentheses). They can also be printed at a different size by overriding AccidentalCautionary’s font-size property.

modern-voice

This rule is used for multivoice accidentals to be read both by musicians playing one voice and musicians playing all voices. Accidentals are typeset for each voice, but they are canceled across voices in the same Staff. Hence, the \( a \) in the last measure is canceled because the previous cancellation was in a different voice, and the \( d \) in the lower staff is canceled because of the accidental in a different voice in the previous measure:
modern-voice-cautionary

This rule is the same as modern-voice, but with the extra accidentals (the ones not typeset by voice) typeset as cautionaries. Even though all accidentals typeset by default are typeset with this rule, some of them are typeset as cautionaries.

![Example of modern-voice-cautionary notation](image)

piano

This rule reflects twentieth-century practice for piano notation. Its behavior is very similar to modern style, but here accidentals also get canceled across the staves in the same GrandStaff or PianoStaff, hence all the cancellations of the final notes.

This accidental style applies to the current GrandStaff or PianoStaff by default.

![Example of piano notation](image)

piano-cautionary

This is the same as piano but with the extra accidentals typeset as cautionaries.

![Example of piano-cautionary notation](image)

choral

This rule is a combination of the modern-voice and the piano style. It shows all accidentals required for singers that only follow their own voice, as well as additional accidentals for readers that follow all voices of an entire ChoirStaff simultaneously.

This accidental style applies to the current ChoirStaff by default.

![Example of choral notation](image)

choral-cautionary

This is the same as choral but with the extra accidentals typeset as cautionaries.

![Example of choral-cautionary notation](image)
This rule reproduces a common practice in contemporary music: accidentals are printed like with modern, but they are printed again if the same note appears later in the same measure – except if the note is immediately repeated.

This rule is similar to neo-modern, but the ‘extra’ accidentals are printed as cautionary accidentals (with parentheses). They can also be printed at a different size by overriding AccidentalCautionary’s font-size property.

This rule is used for multivoice accidentals to be read both by musicians playing one voice and musicians playing all voices. Accidentals are typeset for each voice as with neo-modern, but they are canceled across voices in the same Staff.

This rule is similar to neo-modern-voice, but the extra accidentals are printed as cautionary accidentals.
dodecaphonic

This rule reflects a practice introduced by composers at the beginning of the 20th century, in an attempt to abolish the hierarchy between natural and non-natural notes. With this style, every note gets an accidental sign, including natural signs.

![Dodecaphonic notation example](image1)

**dodecaphonic-no-repeat**

Like with the dodecaphonic accidental style every note gets an accidental sign by default, but accidentals are suppressed for pitches immediately repeated within the same staff.

![Dodecaphonic-no-repeat notation example](image2)

**dodecaphonic-first**

Similar to the dodecaphonic accidental style every pitch gets an accidental sign, but only the first time it is encountered in a measure. Accidentals are only remembered for the actual octave but throughout voices.

![Dodecaphonic-first notation example](image3)

**teaching**

This rule is intended for students, and makes it easy to create scale sheets with automatically created cautionary accidentals. Accidentals are printed like with modern, but cautionary accidentals are added for all sharp or flat tones specified by the key signature, except if the note is immediately repeated.

![Teaching notation example](image4)

**no-reset**

This is the same as default but with accidentals lasting ‘forever’ and not only within the same measure:
This is the opposite of no-reset: Accidental are not remembered at all – and hence all accidentals are typeset relative to the key signature, regardless of what came before in the music.

See also

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “Accidental” in Internals Reference, Section “Accidental_ engraver” in Internals Reference, Section “GrandStaff” in Internals Reference, Section “PianoStaff” in Internals Reference, Section “Staff” in Internals Reference, Section “AccidentalSuggestion” in Internals Reference, Section “AccidentalPlacement” in Internals Reference, Section “accidental-suggestion-interface” in Internals Reference.

Known issues and warnings

Simultaneous notes are not considered in the automatic determination of accidentals; only previous notes and the key signature are considered. Forcing accidentals with \! or \? may be required when the same note name occurs simultaneously with different alterations, as in '<f! fis!>'.

Cautionary cancellation of accidentals is done by looking at previous measure. However, in the alternative block following a \repeat volta N section, one would expect the cancellation being calculated using the previous played measure, not previous printed measure. In the following example, the natural c in the second alternative does not need a natural sign:

The following work-around can be used: define a function that locally changes the accidental style to forget:

```ly
\repeat volta 2 { 
  c'2
}
```
Ambitus

The term *ambitus* (pl. ambitus) denotes a range of pitches for a given voice in a part of music. It may also denote the pitch range that a musical instrument is capable of playing. Ambitus are printed on vocal parts so that performers can easily determine if it matches their capabilities.

Ambitus are denoted at the beginning of a piece near the initial clef. The range is graphically specified by two note heads that represent the lowest and highest pitches. Accidentals are only printed if they are not part of the key signature.

**Selected Snippets**

*Adding ambitus per voice*

Ambitus can be added per voice. In this case, the ambitus must be moved manually to prevent collisions.

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \override Ambitus.X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
```

Ambitus with multiple voices

Adding the Ambitus_engraver to the Staff context creates a single ambitus per staff, even in the case of staves with multiple voices.

```
\new Staff \with {
  \consists "Ambitus_engraver"
}
<<
\new Voice \relative c'' { 
  \voiceOne 
  c4 a d e 
  f1 
}
\new Voice \relative c' { 
  \voiceTwo 
  es4 f g as 
  b1 
}
>>
```

Changing the ambitus gap

It is possible to change the default gap between the ambitus noteheads and the line joining them.

```
\layout { 
  \context { 
    \Voice 
    \consists "Ambitus_engraver"
  }
}
\new Staff { 
  \time 2/4 
  % Default setting 
  c'4 g'' 
}
```
\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1.5
  c'4 g''
}

See also

Snippets: Section “Pitches” in Snippets.

Known issues and warnings

There is no collision handling in the case of multiple per-voice ambitus.

1.1.4 Note heads

This section suggests ways of altering note heads.
Special note heads

The appearance of note heads may be altered:
\begin{verbatim}
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
  \override NoteHead.style = #'harmonic
  a b
  \revert NoteHead.style
  c4 d e f
}
\end{verbatim}

To see all note head styles, see Section A.9 [Note head styles], page 705.

The cross style is used to represent a variety of musical intentions. The following generic predefined commands modify the note head in both staff and tablature contexts and can be used to represent any musical meaning:
\begin{verbatim}
\relative { 
  c''4 b 
  \xNotesOn
  a b c4 b 
  \xNotesOff
  c4 d 
}
\end{verbatim}

The music function form of this predefined command may be used inside and outside chords to generate crossed note heads in both staff and tablature contexts:
\begin{verbatim}
\relative { 
  c''4 b 
  \xNote { e f }
  c b < g \xNote c f > b 
}
\end{verbatim}

As synonyms for \xNote, \xNotesOn and \xNotesOff, \deadNote, \deadNotesOn and \deadNotesOff can be used. The term dead note is commonly used by guitarists.

There is also a similar shorthand for diamond shapes:
\begin{verbatim}
\relative c'' { 

\end{verbatim}
Predefined commands
\harmonic, \xNotesOn, \xNotesOff, \xNote.

See also
Snippets: Section “Pitches” in Snippets.
Notation Reference: Section A.9 [Note head styles], page 705, [Chorded notes], page 168, [Indicating harmonics and dampened notes], page 398.

Easy notation note heads
The ‘easy play’ note head includes a note name inside the head. It is used in music for beginners. To make the letters readable, it should be printed in a large font size. To print with a larger font, see Section 4.2.2 [Setting the staff size], page 559.

```
#(set-global-staff-size 26)
\relative c' {
 \easyHeadsOn
  c2 e4 f
  g1
 \easyHeadsOff
  c,1
}
```

Predefined commands
\easyHeadsOn, \easyHeadsOff.

Selected Snippets

Numbers as easy note heads
Easy notation note heads use the note-names property of the NoteHead object to determine what appears inside the note head. By overriding this property, it is possible to print numbers representing the scale-degree.

A simple engraver can be created to do this for every note head object it sees.

```
#(define Ez_numbers_engraver
```
(make-engraver
  (acknowledgers
   ((note-head-interface engraver grob source-engraver)
    (let* ((context (ly:translator-context engraver))
           (tonic-pitch (ly:context-property context 'tonic))
           (tonic-name (ly:pitch-notename tonic-pitch))
           (grob-pitch
            (ly:event-property (event-cause grob) 'pitch))
           (grob-name (ly:pitch-notename grob-pitch))
           (delta (modulo (- grob-name tonic-name) 7))
           (note-names
            (make-vector 7 (number->string (1+ delta)))))
        (ly:grob-set-property! grob 'note-names note-names)))))

#(set-global-staff-size 26)

\layout {
  \ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\relative c' {
  \easyHeadsOn
  c4 d e f
g4 a b c \break

  \key a \major
  a,4 b cis d
e4 fis gis a \break

  \key d \dorian
d,4 e f g
  a4 b c d
}

\times 6/8
\accidentals.sharp/\accidentals.sharp/\accidentals.sharp
\times 3/4
\accidentals.natural/\accidentals.natural/\accidentals.natural
\times 5/8
\clefs.G\clefs.G\clefs.G

\times 3/4
\clefs.G\clefs.G
\times 5/8
\clefs.G\clefs.G
See also

Notation Reference: Section 4.2.2 [Setting the staff size], page 559.
Snippets: Section “Pitches” in Snippets.
Internals Reference: Section “note-event” in Internals Reference, Section “Note_heads_ engraver” in Internals Reference, Section “NoteHead” in Internals Reference, Section “note-head-interface” in Internals Reference.

Shape note heads

In shape note head notation, the shape of the note head corresponds to the harmonic function of a note in the scale. This notation was popular in nineteenth-century American song books. Shape note heads can be produced in Sacred Harp, Southern Harmony, Funk (Harmonia Sacra), Walker, and Aiken (Christian Harmony) styles:

\relative c' { 
  \aikenHeads 
  c, d e f g2 a b1 c \break  
  \aikenThinHeads 
  c,4 d e f g2 a b1 c \break  
  \sacredHarpHeads 
  c,4 d e f g2 a b1 c \break  
  \southernHarmonyHeads 
  c,4 d e f g2 a b1 c \break  
  \funkHeads 
  c,4 d e f g2 a b1 c \break  
  \walkerHeads 
  c,4 d e f g2 a b1 c \break 
}

\relative c' { 
  \aikenHeads 
  c, d e f g2 a b1 c \break  
  \aikenThinHeads 
  c,4 d e f g2 a b1 c \break  
  \sacredHarpHeads 
  c,4 d e f g2 a b1 c \break  
  \southernHarmonyHeads 
  c,4 d e f g2 a b1 c \break  
  \funkHeads 
  c,4 d e f g2 a b1 c \break  
  \walkerHeads 
  c,4 d e f g2 a b1 c \break 
}
Shapes are typeset according to the step in the scale, where the base of the scale is determined by the `\key` command. When writing in a minor key, the scale step can be determined from the relative major:

```
\relative c' {  
\key a \minor  
aikenHeads  
a b c d e2 f g1 a \break  
aikenHeadsMinor  
a,4 b c d e2 f g1 a \break  
aikenThinHeadsMinor  
a,4 b c d e2 f g1 a \break  
sacredHarpHeadsMinor  
a,2 b c d \break  
southernHarmonyHeadsMinor  
a2 b c d \break  
funkHeadsMinor  
a2 b c d \break  
walkerHeadsMinor  
a2 b c d \break
}
```

![Musical notation examples](image-url)
Predefined commands
\aikenHeads, \aikenHeadsMinor, \aikenThinHeads, \aikenThinHeadsMinor, \funkHeads, \funkHeadsMinor, \sacredHarpHeads, \sacredHarpHeadsMinor, \southernHarmonyHeads, \southernHarmonyHeadsMinor, \walkerHeads, \walkerHeadsMinor.

Selected Snippets

Aiken head thin variant noteheads

Aiken head white notes get harder to read at smaller staff sizes, especially with ledger lines. Losing interior white space makes them appear as quarter notes.
\score {  
  \aikenHeads  
  c''2 a' c' a  
  \% Switch to thin-variant noteheads  
  \set shapeNoteStyles = ##(doThin reThin miThin faThin sol laThin tiThin)  
  c'' a' c' a  
}  
\% END EXAMPLE

Applying note head styles depending on the step of the scale

The shapeNoteStyles property can be used to define various note head styles for each step of the scale (as set by the key signature or the tonic property). This property requires a set of symbols, which can be purely arbitrary (geometrical expressions such as triangle, cross, and xcircle are allowed) or based on old American engraving tradition (some latin note names are also allowed).

That said, to imitate old American song books, there are several predefined note head styles available through shortcut commands such as \aikenHeads or \sacredHarpHeads.

This example shows different ways to obtain shape note heads, and demonstrates the ability to transpose a melody without losing the correspondence between harmonic functions and note head styles.

fragment = {  
  \key c \major  
  c2 d  
  e2 f  
  g2 a  
  b2 c  
}

\new Staff {  
  \transpose c d  
  \relative c' {  
    \set shapeNoteStyles = ##(do re mi fa  
      #f la ti)  
  }  
  \fragment
To see all note head styles, see Section A.9 [Note head styles], page 705.

See also
Snippets: Section “Pitches” in Snippets.
Notation Reference: Section A.9 [Note head styles], page 705.
Internals Reference: Section “note-event” in Internals Reference, Section “Note_heads engraver” in Internals Reference, Section “NoteHead” in Internals Reference, Section “note-head-interface” in Internals Reference.

Improvisation
Improvisation is sometimes denoted with slashed note heads, where the performer may choose any pitch but should play the specified rhythm. Such note heads can be created:

\new Voice \with { \consists "Pitch_squash engraver" } \relative { e''8 e g a a16( bes) a8 g \improvisationOn e8 \improvisationOff 2 \improvisationOff a16( bes) a8 g e }

Predefined commands
\improvisationOn, \improvisationOff.
See also

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “Pitch_squash_ engraver” in Internals Reference, Section “Voice” in Internals Reference, Section “RhythmicStaff” in Internals Reference.

1.2 Rhythms

This section discusses rhythms, rests, durations, beaming and bars.

1.2.1 Writing rhythms
Durations

The durations of notes are entered using numbers and dots. The number entered is based on the reciprocal value of the length of the note. For example, a quarter note is designated using the numerical value of 4 as it is a 1/4 note, a half note using 2, an eighth using 8 and so on. Durations as short as 1024 notes can be entered but shorter values, while possible, can only be entered as beamed notes. Also see Section 1.2.4 [Beams], page 84.

For notes longer than a whole use the \longa – double breve – and \breve commands. A note with the duration of a quadruple breve is possible using the \maxima command but is only supported within ancient music notation. See Section 2.9 [Ancient notation], page 446.

\relative {
  \time 8/1
  c'' \longa c \breve 1 c2
  c4 c8 c16 c32 c64 c128 c128
}

Here are the same durations with automatic beaming turned off.

\relative {
  \time 8/1
  \autoBeamOff
  c'' \longa c \breve 1 c2
  c4 c8 c16 c32 c64 c128 c128
}

Isolated durations – durations without a pitch – that occur within a music sequence will take their pitch from the preceding note or chord.

\relative {
  \time 8/1
  c'' \longa \breve 1 2
  4 8 16 32 64 128 128
}

Isolated pitches – pitches without a duration – that occur within a music sequence will take their duration from the preceding note or chord. If there is no preceding duration, then default for the note is always 4, a quarter note.

\relative { a' a a2 a a4 a a1 a }
Place a dot (\.) after the duration to obtain ‘dotted’ note lengths. Double-dotted notes are specified by appending two dots, and so on.

\relative { a'4 b c4. b8 a4. b4.. c8. }

To avoid clashing with staff lines, dots on notes are normally moved up. In polyphonic situations however, they can be placed, manually, above or below the staff as required. See Section 5.4.2 [Direction and placement], page 634.

Some note durations cannot be represented using just numbers and dots but only by tying two or more notes together. See [Ties], page 55.

To specify durations that align the syllables of lyrics and notes together see Section 2.1 [Vocal music], page 274.

Notes can also be spaced proportionately to their duration, see Section 4.5.5 [Proportional notation], page 587.

Predefined commands
\autoBeamOn, \autoBeamOff, \dotsUp, \dotsDown, \dotsNeutral.

Selected Snippets

**Alternative breve notes**

Breve notes are also available with two vertical lines on each side of the notehead instead of one line and in baroque style.

\relative c'' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}

*Changing the number of augmentation dots per note*

The number of augmentation dots on a single note can be changed independently of the dots placed after the note.

\relative c' {
  c4.. a16 r2 |
  \override Dots.dot-count = #4
  c4.. a16 r2 |
  \override Dots.dot-count = #0
Chapter 1: Musical notation

\begin{music}
c4.. a16 r2 |
\revert Dots.dot-count
\end{music}

\begin{music}
c4.. a16 r2 |
\end{music}

See also

Notation Reference: Section 1.2.4 [Beams], page 84, [Ties], page 55, [Stems], page 234, Section 1.2.1 [Writing rhythms], page 46, Section 1.2.2 [Writing rests], page 59, Section 2.1 [Vocal music], page 274, Section 2.9 [Ancient notation], page 446, Section 4.5.5 [Proportional notation], page 587.

Snippets: Section “Rhythms” in *Snippets*.


Known issues and warnings
While there is no fundamental limit to rest durations (longest or shortest), there is a limit to the number of glyphs possible so only rests between 1024 and \maxima may be printed.

Tuplets
Tuplets are made from a music expression with the \tuplet command, multiplying the speed of the music expression by a fraction:

\begin{music}
\relative {
  a'2 \tuplet 3/2 { b4 4 4 }
  c4 c \tuplet 3/2 { b4 a g }
}\end{music}

When entering long passages of tuplets, having to write a separate \tuplet command for each group is inconvenient. It is possible to specify the duration of one tuplet group directly before the music in order to have the tuplets grouped automatically:

\begin{music}
\relative {
  g'2 r8 \tuplet 3/2 8 { cis16 d e e f g g f e }
}\end{music}
Tuplet brackets may be manually placed above or below the staff:
\relative {
    \tupletUp \tuplet 3/2 { c''8 d e }
    \tupletNeutral \tuplet 3/2 { c8 d e }
    \tupletDown \tuplet 3/2 { f,8 g a }
    \tupletNeutral \tuplet 3/2 { f8 g a }
}

Tuplets may be nested:
\relative {
    \autoBeamOff
    c''4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4
}

Modifying nested tuplets which begin at the same musical moment must be done with \tweak; see Section 5.3.4 [The tweak command], page 623.

Tuplet brackets may be replaced with slurs, as is preferred in many older editions:
\relative {
    \tuplet 3/2 4 {
        \override TupletBracket.tuplet-slur = ##t
        c'4 e8 d4 f8
        \override TupletBracket.bracket-visibility = ##t
        e f g f e d
    } c1
}

By default, a bracket is only printed if all of the notes it spans are not beamed together; in some cases (for example with slurs, as in the example above) it may be preferrable to change that behavior, through the bracket-visibility property as detailed in one of the following snippets.

More generally, either or both the TupletBracket and TupletNumber objects may be hidden or shown as explained in Section 5.4.7 [Visibility of objects], page 641; however, a more flexible way of modifying the duration of notes without printing a tuplet bracket is also introduced in [Scaling durations], page 54.

Predefined commands
\tupletUp, \tupletDown, \tupletNeutral.
Selected Snippets

Entering several tuplets using only one \tuplet command

The property \tupletSpannerDuration sets how long each of the tuplets contained within
the brackets after \tuplet should last. Many consecutive tuplets can then be placed within a
single \tuplet expression, thus saving typing.

There are several ways to set \tupletSpannerDuration. The command \tupletSpan sets it
to a given duration, and clears it when instead of a duration \default is specified. Another
way is to use an optional argument with \tuplet.

\relative c' {\time 2/4 \tupletSpan 4 \tuplet 3/2 { c8^"\tupletSpan 4" c c c c } \tupletSpan \default \tuplet 3/2 { c8^"\tupletSpan \default" c c c c } \tuplet 3/2 4 { c8^"\tuplet 3/2 4 {...}" c c c c }\}

Changing the tuplet number

By default, only the numerator of the tuplet number is printed over the tuplet bracket, i.e.,
the numerator of the argument to the \tuplet command.

Alternatively, num:den of the tuplet number may be printed, or the tuplet number may be
suppressed altogether.

\relative c'' {\once \override TupletNumber.text = #tuplet-number::calc-fraction-text \tuplet 3/2 { c8 c c } \omit TupletNumber \tuplet 3/2 { c8 c c }\}

Non-default tuplet numbers

LilyPond also provides formatting functions to print tuplet numbers different than the actual
fraction, as well as to append a note value to the tuplet number or tuplet fraction.

\relative c'' {\once \override TupletNumber.text = #(tuplet-number::non-default-tuplet-denominator-text 7) \tuplet 3/2 { c4. c4. c4. c4. } \once \override TupletNumber.text = #(tuplet-number::non-default-tuplet-fraction-text 12 7)
Controlling tuplet bracket visibility

The default behavior of tuplet-bracket visibility is to print a bracket unless there is a beam of the same length as the tuplet. To control the visibility of tuplet brackets, set the property `bracket-visibility` to either `#t` (always print a bracket), `if-no-beam` (only print a bracket if there is no beam, which is the default behavior), or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `TupletBracket` object altogether from the printed output.

```latex
\music = \relative c' {
\tuplet 3/2 { c16[ d e ] f8}
\tuplet 3/2 { c8 d e }
\tuplet 3/2 { c4 d e }
}

\new Voice {
\relative c' {
<< \music s4"default" >>
\override TupletBracket.bracket-visibility = #'if-no-beam
<< \music s4"if-no-beam" >>
\override TupletBracket.bracket-visibility = ##t
<< \music s4"#t" >>
\override TupletBracket.bracket-visibility = ##f
<< \music s4"#f" >>
```
Permitting line breaks within beamed tuplets

This artificial example shows how both manual and automatic line breaks may be permitted to within a beamed tuplet. Note that such off-beat tuplets have to be beamed manually.

\layout {
\context {
\Voice
% Permit line breaks within tuplets
\remove "Forbid_line_break_engraver"
% Allow beams to be broken at line breaks
\override Beam.breakable = ##t
}
}
\relative c' {
a8
\repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
% Insert a manual line break within a tuplet
\tuplet 3/2 { c[ b \bar "" \break a] }
\repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
c8
}
The duration of single notes, rests or chords may be multiplied by a fraction \( \frac{N}{M} \) by appending \(*\frac{N}{M}\) (or \(*N\) if \(M\) is 1) to the duration. Factors may also be added by using Scheme expressions evaluating to a number or musical length like \(*\text{(ly:music-length music)}\). This is convenient for scaling a duration of ‘1’ to let a note or multi-measure rest stretch to a length derived from a music variable.

Adding a factor will not affect the appearance of the notes or rests produced, but the altered duration will be used in calculating the position within the measure and setting the duration in the MIDI output. Multiplying factors may be combined like \(*L*\frac{M}{N}\). Factors are part of the duration: if a duration is not specified for subsequent notes, the default duration taken from the preceding note will include any scaling factor.

In the following example, the first three notes take up exactly two beats, but no triplet bracket is printed.

\[
\text{\% Alter durations to triplets}\n\text{\% Normal durations}\n\text{\% Double the duration of chord}\n\text{\% Duration of quarter, appears like sixteenth}\n\]

The duration of spacer rests may also be modified by a multiplier. This is useful for skipping many measures, e.g., \(s1*23\).

Longer stretches of music may be compressed by a fraction in the same way, as if every note, chord or rest had the fraction as a multiplier. This leaves the appearance of the music unchanged but the internal duration of the notes will be multiplied by the given scale factor, usually \(num/den\). Here is an example showing how music can be compressed and expanded:

\[
\text{\% Alter durations to triplets}\n\text{\% Normal durations}\n\text{\% Double the duration of chord}\n\text{\% Duration of quarter, appears like sixteenth}\n\]

The duration of spacer rests may also be modified by a multiplier. This is useful for skipping many measures, e.g., \(s1*23\).
\time 2/4
\% Normal durations
<c'' a>4 c8 a
\% Scale music by *2/3
\scaleDurations 2/3 {
  <c a f>4. c8 a f
}
\% Scale music by *2
\scaleDurations 2 {
  <c' a>4 c8 b
}

One application of this command is in polyrhythmic notation, see [Polyrhythms], page 77.

See also
Notation Reference: [Tuplets], page 49, [Invisible rests], page 61, [Polyrhythms], page 77.
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings
The calculation of the position within a measure must take into account all the scaling factors applied to the notes within that measure and any fractional carry-out from earlier measures. This calculation is carried out using rational numbers. If an intermediate numerator or denominator in that calculation exceeds 2^{30} the execution and typesetting will stop at that point without indicating an error.

Ties
A tie connects two adjacent notes of the same pitch. The tie in effect extends the duration of a note.

Note: Ties should not be confused with slur, which indicate articulation, or phrasing slurs, which indicate musical phrasing. A tie is just a way of extending a note duration, similar to the augmentation dot.

A tie is entered by appending a tilde symbol (~) to the first of each pair of notes being tied. This indicates that the note should be tied to the following note, which must be at the same pitch.

\{ a'2~ 4~ 16 r r8 \}

Ties can make use of the ‘last explicit pitch’ interpretation of isolated durations:

\{ a'2~ 4~ 16 r r8 \}
Ties are used either when the note crosses a bar line, or when dots cannot be used to denote the rhythm. Ties should also be used when note values cross larger subdivisions of the measure:

\relative { 
  r8 c'4~ 4 r4 | 
  r8"not" c2~ 8 r4 } 

If you need to tie many notes across bar lines, it may be easier to use automatic note splitting, see [Automatic note splitting], page 80. This mechanism automatically splits long notes, and ties them across bar lines.

When a tie is applied to a chord, all note heads whose pitches match are connected. When no note heads match, no ties will be created. Chords may be partially tied by placing the ties inside the chord.

\relative c' { 
  <c e g>2~ 2 | 
  <c e g>4~ <c e g c> 
  \repeatTie <c~ e g~ b> <c e g b> | 
} 

When a second alternative of a repeat starts with a tied note, you have to specify the repeated tie as follows:

\relative { 
  \repeat volta 2 { c' g \<c e>2~ } 
  \alternative { 
    % First alternative: following note is tied normally 
    { <c e>2. r4 } 
    % Second alternative: following note has a repeated tie 
    { <c e>2\repeatTie d4 c } 
  } 
} 

\laissezVibrer

\laissezVibrer

L.v. ties (laissez vibrer) indicate that notes must not be damped at the end. It is used in notation for piano, harp and other string and percussion instruments. They can be entered as follows:

<c' f' g'>1\laissezVibrer
Ties may be made to curve up or down manually; see Section 5.4.2 [Direction and placement], page 634.

Ties may be made dashed, dotted, or a combination of solid and dashed.

\relative c' {
  \tieDotted
c2~ 2
  \tieDashed
c2~ 2
  \tieHalfDashed
c2~ 2
  \tieHalfSolid
c2~ 2
  \tieSolid
c2~ 2
}

Custom dash patterns can be specified:

\relative c' {
  \tieDashPattern #0.3 #0.75
c2~ 2
  \tieDashPattern #0.7 #1.5
c2~ 2
  \tieSolid
c2~ 2
}

Dash pattern definitions for ties have the same structure as dash pattern definitions for slurs. For more information about complex dash patterns, see [Slurs], page 133.

Override whiteout and layer layout properties of objects that should cause a gap in ties.

\relative {
  \override Tie.layer = #-2
  \override Staff.TimeSignature.layer = #-1
  \override Staff.KeySignature.layer = #-1
  \override Staff.TimeSignature.whiteout = ##t
  \override Staff.KeySignature.whiteout = ##t
  b'2 b~
  \time 3/4
  \key a \major
  b r4
}
Predefined commands
\tieUp, \tieDown, \tieNeutral, \tieDotted, \tieDashed, \tieDashPattern, \tieHalfDashed, \tieHalfSolid, \tieSolid.

Selected Snippets

Using ties with arpeggios

Ties are sometimes used to write out arpeggios. In this case, two tied notes need not be consecutive. This can be achieved by setting the tieWaitForNote property to #t. The same feature is also useful, for example, to tie a tremolo to a chord, but in principle, it can also be used for ordinary consecutive notes.

\relative c' {
    \set tieWaitForNote = ##t
    \grace { c16[ ~ e ~ g] ~ } <c, e g>2
    \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
    e8 ~ c ~ a ~ f ~ <e' c a f>2
    \tieUp
c8 ~ a
    \tieDown
    \tieDotted
g8 ~ c g2
}

Engraving ties manually

Ties may be engraved manually by changing the tie-configuration property of the TieColumn object. The first number indicates the distance from the center of the staff in half staff-spaces, and the second number indicates the direction (1 = up, -1 = down).

Note that LilyPond makes a distinction between exact and inexact values for the first number. If using an exact value (i.e., either an integer or a fraction like (/ 4 5)), the value serves as a rough vertical position that gets further tuned by LilyPond to make the tie avoid staff lines. If using an inexact value like a floating point number, it is taken as the vertical position without further adjustments.

\relative c' {
    <c e g>2 ~ <c e g>
    \override TieColumn.tie-configuration = #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
    <c e g>2 ~ <c e g>
    \override TieColumn.tie-configuration = #'((0 . 1) (-2 . 1) (-4 . 1))
    <c e g>2 ~ <c e g>
}
See also

Notation Reference: [Slurs], page 133, [Automatic note splitting], page 80.
Snippets: Section “Expressive marks” in Snippets, Section “Rhythms” in Snippets.

Known issues and warnings

Switching staves when a tie is active will not produce a slanted tie.
Changing clefs or ottavations during a tie is not really well-defined. In these cases, a slur may be preferable.

1.2.2 Writing rests

Rests are entered as part of the music in music expressions.

Rests

Rests are entered like notes with the note name r. Durations longer than a whole rest use the following predefined commands:

\new Staff {
% These two lines are just to prettify this example
\time 16/1
\omit Staff.TimeSignature
% Print a maxima rest, equal to four breves
r\maxima
% Print a longa rest, equal to two breves
r\longa
% Print a breve rest
r\breve
r1 r2 r4 r8 r16 r32 r64 r128
}

Whole measure rests, centered in the middle of the measure, must be entered as multi-measure rests. They can be used for a single measure as well as many measures and are discussed in [Full measure rests], page 62.

To explicitly specify a rest’s vertical position, write a note followed by \rest. A rest of the duration of the note will be placed at the staff position where the note would appear. This allows for precise manual formatting of polyphonic music, since the automatic rest collision formatter will not move these rests.

\relative { a'4\rest d4\rest }
Selected Snippets

Rest styles
Rests may be used in various styles.

\new Staff \relative c {
   \omit Score.TimeSignature
   \cadenzaOn

\override Staff.Rest.style = #'mensural
r\maxima\markup \typewriter { mensural }
r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
\bar ""
\break

\override Staff.Rest.style = #'neomensural
r\maxima\markup \typewriter { neomensural }
r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
\bar ""
\break

\override Staff.Rest.style = #'classical
r\maxima\markup \typewriter { classical }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
\bar ""
\break

\override Staff.Rest.style = #'z
r\maxima\markup \typewriter { z-style }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
\bar ""
\break

\override Staff.Rest.style = #'default
r\maxima\markup \typewriter { default }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
}

\mensural
\begin{music}
\bar \CMSong \S1 (c)\music
\end{music}
\new Staff \relative G {
\hspace{-1cm}
\begin{music}
\bar \CMSong \S1 (c)\music
\end{music}
\}

\neomensural
\new Staff \relative G {
\hspace{-1cm}
\begin{music}
\bar \CMSong \S1 (c)\music
\end{music}
\}

\classical
\new Staff \relative G {
\hspace{-1cm}
\begin{music}
\bar \CMSong \S1 (c)\music
\end{music}
\}

\z
\new Staff \relative G {
\hspace{-1cm}
\begin{music}
\bar \CMSong \S1 (c)\music
\end{music}
\}
See also


Notation Reference: [Full measure rests], page 62.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “Rest” in Internals Reference.

Known issues and warnings

There is no fundamental limit to rest durations (both in terms of longest and shortest), but the number of glyphs is limited: there are rests from 1024th to maxima (8\times whole).

Invisible rests

An invisible rest (also called a ‘spacer rest’) can be entered like a note with the note name s:

\relative c' {
    c4 c s c |
    s2 c |
}

spacer rests are available only in note mode and chord mode. In other situations, for example, when entering lyrics, the command \skip is used to skip a musical moment. \skip requires an explicit duration, but this is ignored if the lyrics derive their durations from the notes in an associated melody through \addlyrics or \lyricsto.

<<
{ a'2 \skip2 a'2 a'2 }
\new Lyrics {
    \lyricmode {
        foo2 \skip 1 bla2
    }
}
>>

Because \skip is a command, it does not affect the default durations of following notes, unlike s.

<<
{ \repeat unfold 8 { a'4 } }
>
A spacer rest implicitly causes \texttt{Staff} and \texttt{Voice} contexts to be created if none exist, just like notes and rests do:

\begin{verbatim}
\{ s1 s s \}
\end{verbatim}

\verb|\skip| simply skips musical time; it creates no output of any kind.

\begin{verbatim}
\% This is valid input, but does nothing
\{ \skip 1 \skip1 \skip 1 \}
\end{verbatim}

See also

Notation Reference: [Hidden notes], page 231, Section 5.4.7 [Visibility of objects], page 641.
Snippets: Section “Rhythms” in \textit{Snippets}.
Internals Reference: Section “SkipMusic” in \textit{Internals Reference}.

\section*{Full measure rests}

Rests for one or more full measures are entered like notes with the note name uppercase ‘R’. Their duration is entered identically to the duration notation used for notes, including the ability to use duration multipliers, as explained in [Scaling durations], page 54:

\begin{verbatim}
\% Rest measures contracted to single measure
\compressMMRests { 
  R1\times4
  R1\times24
  R1\times4
  b'2""Tutti" b'4 a'4
}
\end{verbatim}

The example above also demonstrates how to compress multiple empty measures, as explained in [Compressing empty measures], page 222.
The duration of a multi-measure rest must always be equal to the length of one or several measures. Therefore, some time signatures require the use of augmentation dots or fractions:

\compressMMRests {
  \time 2/4
  R1 | R2 |
  \time 3/4
  R2. | R2.*2 |
  \time 13/8
  R1*13/8 | R1*13/8*12 |
  \time 10/8
  R4*5*4 |
}

A full-measure rest is printed as either a whole or breve rest, centered in the measure, depending on the time signature.

\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |

Markups can be added to multi-measure rests.

\compressMMRests {
  \time 3/4
  R2.*10^\markup { \italic "ad lib." }
}

\markup { \italic "ad lib." }

Note: Markups and articulations attached to a multi-measure rest are objects of types MultiMeasureRestText and MultiMeasureRestScript, not TextScript and Script. Overrides must be directed to the correct object, or they will be ignored. See the following example:

% This fails, as the wrong object name is specified
When a multi-measure rest immediately follows a \partial setting, resulting bar-check warnings may not be displayed.

Predefined commands
\textLengthOn, \textLengthOff, \compressMMRests.

Selected Snippets

Multi-measure rest length control
Multi-measure rests have length according to their total duration which is under the control of MultiMeasureRest.space-increment. Note that the default value is 2.0.

\relative c' {
\compressEmptyMeasures
R1\*2 R1\*4 R1\*64 R1\*16
\override Staff.MultiMeasureRest.space-increment = 2.5
R1\*2 R1\*4 R1\*64 R1\*16
}

Positioning multi-measure rests
Unlike ordinary rests, there is no predefined command to change the staff position of a multi-measure rest symbol of either form by attaching it to a note. However, in polyphonic music multi-measure rests in odd-numbered and even-numbered voices are vertically separated. The positioning of multi-measure rests can be controlled as follows:

\relative c' {
% Multi-measure rests by default are set under the fourth line
R1
% They can be moved using an override
\override MultiMeasureRest.staff-position = #\-
\override MultiMeasureRest.staff-position = #0
\override MultiMeasureRest.staff-position = #2
\override MultiMeasureRest.staff-position = #3
\override MultiMeasureRest.staff-position = #6
\revert MultiMeasureRest.staff-position
\break

\% In two Voices, odd-numbered voices are under the top line
\<< \{ R1 \} \ \{ a1 \} >>
\% Even-numbered voices are under the bottom line
\<< \{ a1 \} \ \{ R1 \} >>
\% Multi-measure rests in both voices remain separate
\<< \{ R1 \} \ \{ R1 \} >>

\% Separating multi-measure rests in more than two voices
\% requires an override
\<< \{ R1 \} \ \{ R1 \} \ \{ R1 \} \ \{ R1 \} \\
\once \override MultiMeasureRest.staff-position = #0
\{ R1 \} >>

\% Using compressed bars in multiple voices requires another override
\% in all voices to avoid multiple instances being printed
\compressMMRests
\<< \\
\revert MultiMeasureRest.direction
\{ R1*3 \} \\
\revert MultiMeasureRest.direction
\{ R1*3 \} >>

Multi-measure rest markup

Markups attached to a multi-measure rest will be centered above or below it. Long markups attached to multi-measure rests do not cause the measure to expand. To expand a multi-measure rest to fit the markup, use an empty chord with an attached markup before the multi-measure rest.
Text attached to a spacer rest in this way is left-aligned to the position where the note would be placed in the measure, but if the measure length is determined by the length of the text, the text will appear to be centered.

\relative c' { 
\compressMMRests { 
\textLengthOn 
<> \markup { \[MAJOR GENERAL\] } 
R1\*19 
<> \markup { \italic { Cue: ... it is yours } } 
<> \markup { A } 
R1\*30 \markup { [MABEL] } 
\textLengthOff 
c4 \markup { CHORUS } d f c 
} 

[MAJOR GENERAL] \hspace{1cm} [MABEL] \hspace{1cm} CHORUS

\text {Cue: ... it is yours}

\text {\hspace{1cm}}

See also


Notation Reference: [Durations], page 47, [Scaling durations], page 54, [Compressing empty measures], page 222, Section 1.8 [Text], page 242, Section 1.8.2 [Formatting text], page 252, [Text scripts], page 245.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “MultiMeasureRest” in Internals Reference, Section “MultiMeasureRestNumber” in Internals Reference, Section “MultiMeasureRestScript” in Internals Reference, Section “MultiMeasureRestText” in Internals Reference.

Known issues and warnings

Fingerings over multi-measure rests (e.g., R1*10-4) may result in the fingering numeral colliding with the bar counter numeral.

There is no way to automatically condense multiple ordinary rests into a single multi-measure rest.

Multi-measure rests do not take part in rest collisions.

1.2.3 Displaying rhythms

Time signature

The time signature is set as follows:

\time 2/4 c''2
\time 3/4 c''2.
Mid-measure time signature changes are covered in [Upbeats], page 74.

Time signatures are printed at the beginning of a piece and whenever the time signature changes. If a change takes place at the end of a line a warning time signature sign is printed there. This default behavior may be changed, see Section 5.4.7 [Visibility of objects], page 641.

\relative c'' {
  \time 2/4
  c2 c
  \break
  c c
  \break
  \time 4/4
  c c c c c
}

The time signature symbol that is used in 2/2 and 4/4 time can be changed to a numeric style:

\relative c'' {
  \% Default style
  \time 4/4 c1
  \time 2/2 c1
  \% Change to numeric style
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  \% Revert to default style
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}

Mensural time signatures are covered in [Mensural time signatures], page 452.

In addition to setting the printed time signature, the \time command also sets the values of the time-signature-based properties baseMoment, beatStructure, and beamExceptions. The predefined default values for these properties can be found in scm/time-signature-settings.scm.
The default value of beatStructure can be overridden in the $time$ command itself by supplying it as the optional first argument:

\score {
\new Staff {
\relative {
\time 2,2,3 7/8
\repeat unfold 7 { c'8 } |
\time 3,2,2 7/8
\repeat unfold 7 { c8 } |
}
}

Alternatively, the default values of all these time-signature-based variables, including baseMoment and beamExceptions, can be set together. The values can be set independently for several different time signatures. The new values take effect when a subsequent $time$ command with the same value of the time signature is executed:

\score {
\new Staff {
\relative c' {
\overrideTimeSignatureSettings
4/4 % timeSignatureFraction
1/4 % baseMomentFraction
3,1 % beatStructure
#()' % beamExceptions
\time 4/4
\repeat unfold 8 { c8 } |
}
}

\overrideTimeSignatureSettings takes four arguments:

1. timeSignatureFraction, a fraction describing the time signature to which these values apply.
2. baseMomentFraction, a fraction containing the numerator and denominator of the basic timing unit for the time signature.
3. beatStructure, a Scheme list indicating the structure of the beats in the measure, in units of the base moment.
4. beamExceptions, an alist containing any beaming rules for the time signature that go beyond ending at every beat, as described in [Setting automatic beam behavior], page 87.

Changed values of default time signature properties can be restored to the original values:

\score {
Different values of default time signature properties can be established for different staves by moving the \texttt{Timing\_translator} and the \texttt{Default\_bar\_line\_engraver} from the \texttt{Score} context to the \texttt{Staff} context.

```latex
\relative {
\repeat unfold 8 { c'8 } |
\overrideTimeSignatureSettings
  4/4 % timeSignatureFraction
  1/4 % baseMomentFraction
  3,1 % beatStructure
  #()' % beamExceptions
\time 4/4
\repeat unfold 8 { c8 } |
\revertTimeSignatureSettings 4/4
\time 4/4
\repeat unfold 8 { c8 } |
}
```

![Musical notation example](noteheads.s2/timesig.C44)

```latex
\score {
\new StaffGroup <<
  \new Staff {
    \overrideTimeSignatureSettings
      4/4 % timeSignatureFraction
      1/4 % baseMomentFraction
      3,1 % beatStructure
      #()' % beamExceptions
    \time 4/4
    \repeat unfold 8 {c''8}
  }
  \new Staff {
    \overrideTimeSignatureSettings
      4/4 % timeSignatureFraction
      1/4 % baseMomentFraction
      1,3 % beatStructure
      #()' % beamExceptions
    \time 4/4
    \repeat unfold 8 {c''8}
  }
>>
\layout {
  \context {
    \Score
    \remove "Timing\_translator"
    \remove "Default\_bar\_line\_engraver"
  }
  \context {
    \Staff
    \consists "Timing\_translator"
  }
```
A further method of changing these time-signature-related variables, which avoids reprinting the time signature at the time of the change, is shown in [Setting automatic beam behavior], page 87.

Predefined commands
\numericTimeSignature, \defaultTimeSignature.

Selected Snippets

*Time signature printing only the numerator as a number (instead of the fraction)*

Sometimes, a time signature should not print the whole fraction (e.g. 7/4), but only the numerator (7 in this case). This can be easily done by using \override Staff.TimeSignature.style = #'single-digit to change the style permanently. By using \revert Staff.TimeSignature.style, this setting can be reversed. To apply the single-digit style to only one time signature, use the \override command and prefix it with a \once.

\relative c' {
    \time 3/4
    c4 c c
    \% Change the style permanently
    \override Staff.TimeSignature.style = #'single-digit
    \time 2/4
    c4 c
    \time 3/4
    c4 c c
    \% Revert to default style:
    \revert Staff.TimeSignature.style
    \time 2/4
    c4 c
    \% single-digit style only for the next time signature
    \once \override Staff.TimeSignature.style = #'single-digit
    \time 5/4
    c4 c c c c
    \time 2/4
    c4 c
}
See also

Music Glossary: Section “time signature” in Music Glossary

Notation Reference: [Mensural time signatures], page 452, [Setting automatic beam behavior], page 87, [Time administration], page 121.

Installed Files: scm/time-signature-settings.scm.

Snippets: Section “Rhythms” in Snippets.


Metronome marks

A basic metronome mark is simple to write:

\relative {
    \tempo 4 = 120
    c'2 d
    e4. d8 c2
}

\relative {
    \tempo 4 = 40 – 46
    c'4. e8 a4 g
    b,2 d4 r
}

\relative {
    \tempo "Allegretto"
    c''4 e d c
    b4. a16 b c4 r4
}

Combining a metronome mark and text will automatically place the metronome mark within parentheses:

\relative {
    \tempo "Allegro" 4 = 160
    g'4 c d e
In general, the text can be any markup object:

\relative {
  \tempo \markup { \italic Faster } 4 = 132
  a'8-. r8 b-. r gis-. r a-. r
}

A parenthesized metronome mark with no textual indication may be written by including an empty string in the input:

\relative {
  \tempo "" 8 = 96
  d''4 g e c
}

In a part for an instrument with long periods of rests (see [Full measure rests], page 62), tempo indications sometimes follow each other closely. The command \markLengthOn provides extra horizontal space to prevent tempo indications from overlapping, and \markLengthOff restores the default behavior of ignoring tempo marks for horizontal spacing.

\compressMMRests {
  \markLengthOn
  \tempo "Molto vivace"
  R1*12
  \tempo "Meno mosso"
  R1*16
  \markLengthOff
  \tempo "Tranquillo"
  R1*20
}

\tempo "Tranquillo"
R1*20
Selected Snippets

Printing metronome and rehearsal marks below the staff

By default, metronome and rehearsal marks are printed above the staff. To place them below the staff simply set the direction property of MetronomeMark or RehearsalMark appropriately.

\layout {
   indent = 0
   ragged-right = ##f
}

{% Metronome marks below the staff
\override Score.MetronomeMark.direction = #DOWN
\tempo 8. = 120
c'1
%

% Rehearsal marks below the staff
\override Score.RehearsalMark.direction = #DOWN
\mark \default
c'1
%

Changing the tempo without a metronome mark

To change the tempo in MIDI output without printing anything, make the metronome mark invisible.

\score {
   \new Staff \relative c' {
      \tempo 4 = 160
c4 e g b
c4 b d c
      \set Score.tempoHideNote = ##t
      \tempo 4 = 96
d,4 fis a cis
d4 cis e d
   }
   \layout { }
   \midi { }
}

Creating metronome marks in markup mode

New metronome marks can be created in markup mode, but they will not change the tempo in MIDI output.

\relative c' {
Chapter 1: Musical notation

\tempo \markup {
  \concat {
    (\smaller \general-align #Y #DOWN \note {16.} \number #1 " = 
      \smaller \general-align #Y #DOWN \note {8} \number #1 )
  }
}
c1
c4 c' c,2

For more details, see Section 1.8.2 [Formatting text], page 252.

See also


Notation Reference: Section 1.8.2 [Formatting text], page 252, Section 3.5 [Creating MIDI output], page 531, [Full measure rests], page 62.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “MetronomeMark” in Internals Reference.

Upbeats

Partial or pick-up measures, such as an anacrusis or an upbeat, are entered using the \partial command:

\partial duration

When \partial is used at the beginning of a score, duration is the length of the music preceding the first bar.

\relative {
  \time 3/4
  \partial 4.
  r4 e'8 | a4 c8 b c4 |
}

When \partial is used after the beginning of a score, duration is the remaining length of the current measure. It does not create a new numbered bar.

\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
Chapter 1: Musical notation

The \texttt{\textbackslash partial} command is \textit{required} when the time signature changes in mid measure, but it may also be used alone.

\begin{verbatim}
\relative c'' {
  c4 d e d \cadenzaOn
c4 c d8[ d d] f4 g4.
}\cadenzaOff
\end{verbatim}

The \texttt{\textbackslash partial} command sets the \texttt{Timing.measurePosition} property, which is a rational number that indicates how much of the measure has passed.

\textbf{See also}

Music Glossary: Section “anacrusis” in \textit{Music Glossary}.
Notation Reference: [Grace notes], page 115.
Snippets: Section “Rhythms” in \textit{Snippets}.
Internal Reference: Section “Timing\_translator” in \textit{Internals Reference}.

\textbf{Unmetered music}

In metered music bar lines are inserted and bar numbers are calculated automatically. In unmetered music (i.e., cadenzas), this is not desirable and can be ‘switched off’ using the command \texttt{\textbackslash cadenzaOn}, then ‘switched back on’ at the appropriate place using \texttt{\cadenzaOff}.

\begin{verbatim}
\relative c' {
  c4 d e d \cadenzaOn
c4 c d8[ d d] f4 g4.
}\cadenzaOff
\end{verbatim}
Bar numbering is resumed at the end of the cadenza.

\relative c'' {
\% Show all bar numbers
\override Score.BarNumber.break-visibility = #all-visible
c4 d e d \cadenzaOn
c4 c d8[ d d ] f4 g4. \cadenzaOff
\bar "|"
d4 e d c
}

Inserting a \bar command within a cadenza does not start a new measure, even if a bar line is printed. So any accidentals – which are usually assumed to remain in force until the end of the measure – will still be valid after the bar line printed by \bar. If subsequent accidentals should be printed, forced accidentals or reminder accidentals need to be inserted manually, see [Accidentals], page 6.

\relative c' {
c4 d e d \cadenzaOn
cis4 d cis d \bar "|
% First cis is printed without alteration even if it’s after a \bar
cis4 d cis! d \cadenzaOff
\bar "|"
}

Automatic beaming is disabled by \cadenzaOn. Therefore, all beaming in cadenzas must be entered manually. See [Manual beams], page 96.

\relative {
\repeat unfold 8 { c''8 }
\cadenzaOn
cis8 c c c c \bar"|
c8 c c
}
These predefined commands affect all staves in the score, even when placed in just one Voice context. To change this, move the Timing_translator from the Score context to the Staff context. See [Polymeric notation], page 77.

Predefined commands
\cadenzaOn, \cadenzaOff.

See also
Notation Reference: Section 5.4.7 [Visibility of objects], page 641, [Polymeric notation], page 77, [Manual beams], page 96, [Accidentals], page 6.
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings
Automatic line and page breaks are inserted only at bar lines, so ‘invisible’ bar lines will need to be inserted manually in long stretches of unmetered music to permit breaking:
\bar

Polymetric notation
Polymetric notation is supported explicitly or by manually modifying the visible time signature symbol and/or scaling note durations.

Different time signatures with equal-length measures
Set a common time signature for each staff, and set the timeSignatureFraction to the desired fraction. Then use the \scaleDurations function to scale the durations of the notes in each staff to the common time signature.

In the following example, music with the time signatures of 3/4, 9/8 and 10/8 are used in parallel. In the second staff, shown durations are multiplied by 2/3 (because 2/3 * 9/8 = 3/4) and in the third staff, the shown durations are multiplied by 3/5 (because 3/5 * 10/8 = 3/4). It may be necessary to insert beams manually, as the duration scaling will affect the autobeaming rules.

\relative <<
\new Staff {
\time 3/4
   c'4 c c c | c4 c c c |
}
\new Staff {
   \time 3/4
   \set Staff.timeSignatureFraction = 9/8
   \scaleDurations 2/3
   \repeat unfold 6 { c8 [ c c ] }
}
Different time signatures with unequal-length measures

Each staff can be given its own independent time signature by moving the Timing_translator and the Default_bar_line_ engraver to the Staff context.

\layout {
\context {
\Score
\remove "Timing_translator"
\remove "Default_bar_line_ engraver"
}
\context {
\Staff
\consists "Timing_translator"
\consists "Default_bar_line_ engraver"
}
}

% Now each staff has its own time signature.

\relative <<
\new Staff {
\time 3/4
c'4 c c |
c4 c c |
}
\new Staff {
\time 2/4
c4 c |
c4 c |
c4 c |
}


\new Staff {
  \time 3/8
  c4. |
  c8 c c |
  c4. |
  c8 c c |
}

\CompoundMeter #'(list of lists)

The simplest construction is a single list, where the last number indicates the bottom number of the time signature and those that come before it, the top numbers.

\relative {
  \CompoundMeter #'((2 2 2 8))
  \repeat unfold 6 c'8 \repeat unfold 12 c16
}

More complex meters can be constructed using additional lists. Also, automatic beaming settings will be adjusted depending on the values.

\relative {
  \CompoundMeter #'((1 4) (3 8))
  \repeat unfold 5 c'8 \repeat unfold 10 c16
}

\relative {
  \CompoundMeter #'((1 2 3 8) (3 4))
  \repeat unfold 12 c'8
}

\CompoundMeter
See also


Notation Reference: [Automatic beams], page 84, [Manual beams], page 96, [Time signature], page 66, [Scaling durations], page 54.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “TimeSignature” in Internals Reference, Section “Timing_translator” in Internals Reference, Section “Default_bar_line_engraver” in Internals Reference, Section “Staff” in Internals Reference.

Known issues and warnings

Although notes that occur at the same moment in each of the different staves will be placed at the same horizontal location, bar lines (in each staff) may cause inconsistent spacing within each of the different time signatures.

Using a midi block with polymetric notation may cause unexpected barcheck warnings. In this case move the Timing_translator from the Score context to the Staff context within the midiblock.

\midi {
\context {
\Score
\remove "Timing_translator"
}
\context {
\Staff
\consists "Timing_translator"
}
}

Automatic note splitting

Long notes which overrun bar lines can be converted automatically to tied notes. This is done by replacing the Note_heads_engraver with the Completion_heads_engraver. Similarly, long rests which overrun bar lines are split automatically by replacing the Rest_engraver with the Completion_rest_engraver. In the following example, notes and rests crossing the bar lines are split, notes are also tied.

\new Voice \with {
\remove "Note_heads_engraver"
\consists "Completion_heads_engraver"
\remove "Rest_engraver"
\consists "Completion_rest_engraver"
}
\relative {
\new TimeSignature \with {
\clef "treble"
\time 3/4
}
c'2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2
}
These engravers split all running notes and rests at the bar line, and inserts ties for notes. One of its uses is to debug complex scores: if the measures are not entirely filled, then the ties show exactly how much each measure is off.

The property `completionUnit` sets a preferred duration for the split notes.

\[\text{new Voice \{ \remove "Note_heads_engraver" \consists "Completion_heads_engraver" \} \relative { \time 9/8 g\breve. d''4. \bar "||" \set completionUnit = #(ly:make-moment 3 8) g\breve. d4. }\]

These engravers split notes with scaled duration, such as those in tuplets, into notes with the same scale-factor as in the input note.

\[\text{new Voice \{ \remove "Note_heads_engraver" \consists "Completion_heads_engraver" \} \relative { \time 2/4 r4 \tuplet 3/2 \{g'4 a b\} \scaleDurations 2/3 \{g a b\} g4*2/3 a b \tuplet 3/2 \{g4 a b\} r4 }\]

See also

Music Glossary: Section “tie” in Music Glossary
Snippets: Section “Rhythms” in Snippets.
Internals Reference: Section “Note_heads_engraver” in Internals Reference, Section “Completion_heads_engraver” in Internals Reference, Section “Rest_engraver” in Internals Reference, Section “Completion_rest_engraver” in Internals Reference, Section “Forbid_line_break_engraver” in Internals Reference.

Known issues and warnings

For consistency with previous behavior, notes and rests with duration longer than a measure, such as c1*2, are split into notes without any scale factor, \{ c1 c1 \}. The property `completionFactor` controls this behavior, and setting it to `#f` cause split notes and rests to have the scale factor of the input durations.
Showing melody rhythms

Sometimes you might want to show only the rhythm of a melody. This can be done with the rhythmic staff. All pitches of notes on such a staff are squashed, and the staff itself has a single line

\begin{verbatim}
\new RhythmicStaff {
    \new Voice = "myRhythm" \relative {
        \time 4/4
        c'4 e8 f g2
        r4 g g f
        g1
    }
}
\new Lyrics {
    \lyricsto "myRhythm" {
        This is my song
        I like to sing
    }
}
\end{verbatim}

\[\text{This is my song} \quad \text{I like to sing}\]

Guitar chord charts often show the strumming rhythms. This can be done with the Pitch_squash_engraver and \improvisationOn.

\begin{verbatim}
\new ChordNames {
    \chordmode {
        c1 f g c
    }
}
\new Voice \with {
    \consists "Pitch_squash_engraver"
} \relative c' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
}
\end{verbatim}

\[\text{C} \quad \text{F} \quad \text{G} \quad \text{C}\]

Music containing chords can also be used as input to RhythmicStaff and for use with the Pitch_squash_engraver if the chords are first reduced to single notes with the \reduceChords music function:
Predefined commands
\improvisationOn, \improvisationOff, \reduceChords.

Selected Snippets

\improvisationOn, \improvisationOff, \reduceChords.

Guitar strum rhythms
For guitar music, it is possible to show strum rhythms, along with melody notes, chord names and fret diagrams.

\include "predefined-guitar-fretboards.ly"

\new ChordNames {
  \chordmode {
    c1 | f | g | c
  }
}
\new FretBoards {
  \chordmode {
    c1 | f | g | c
  }
}
\new Voice \with {
  \consists "Pitch_squash_ engraver"
} {
  \relative c' {\improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c

    } }\new Voice = "melody" {\relative c' {
    c2 e4 e4
    f2. r4
    g2. a4
    e4 c2.}
\new Lyrics {
\lyricsto "melody" {
This is my song.
I like to sing.
}
}

See also
Snippets: Section “Rhythms” in Snippets.
Internals Reference: Section “RhythmicStaff” in Internals Reference, Section “Pitch_squash_engraver” in Internals Reference.

1.2.4 Beams

Automatic beams
By default, beams are inserted automatically:

\relative c'' {
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
}

\relative c' {
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
}
If these automatic decisions are not satisfactory, beaming can be entered explicitly; see [Manual beams], page 96. Beams must be entered manually if beams are to be extended over rests.

If automatic beaming is not required, it may be turned off with `\autoBeamOff` and on with `\autoBeamOn`:

```
\relative c' {
  c4 c8 c8. c16 c8 \autoBeamOff
  c4 c8 c8. c16 c8. \autoBeamOn
  c16 c8
}
```

Note: If beams are used to indicate melismata in songs, then automatic beaming should be switched off with `\autoBeamOff` and the beams indicated manually. Using `\partCombine` with `\autoBeamOff` can produce unintended results. See the snippets for more information.

Beaming patterns that differ from the automatic defaults can be created; see [Setting automatic beam behavior], page 87.

**Predefined commands**

```
\autoBeamOff, \autoBeamOn.
```

**Selected Snippets**

**Beams across line breaks**

Line breaks are normally forbidden when beams cross bar lines. This behavior can be changed as shown:

```
\relative c'' {
  \override Beam.breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
    c8] c[ c] c[ c] c[ c] c
}
```

**Changing beam knee gap**

Kneed beams are inserted automatically when a large gap is detected between the note heads. This behavior can be tuned through the `auto-knee-gap` property. A kneed beam is drawn if the gap is larger than the value of `auto-knee-gap` plus the width of the beam object (which
depends on the duration of the notes and the slope of the beam). By default auto-knee-gap is set to 5.5 staff spaces.

\begin{verbatim}
{ 
  f8 f''8 f8 f''8 
  \override Beam.auto-knee-gap = #6 
  f8 f''8 f8 f''8 
}
\end{verbatim}

\textit{Partcombine and autoBeamOff}

The function of \texttt{\autoBeamOff} when used with \texttt{\partCombine} can be difficult to understand. It may be preferable to use

\begin{verbatim}
\set Staff.autoBeaming = ##f
\end{verbatim}

instead, to ensure that autobeaming will be turned off for the entire staff.

\texttt{\partCombine} apparently works with 3 voices – stem up single, stem down single, stem up combined.

An \texttt{\autoBeamOff} call in the first argument to partcombine will apply to the voice that is active at the time the call is processed, either stem up single or stem up combined. An \texttt{\autoBeamOff} call in the second argument will apply to the voice that is stem down single.

In order to use \texttt{\autoBeamOff} to stop all autobeaming when used with \texttt{\partCombine}, it will be necessary to use three calls to \texttt{\autoBeamOff}.

\begin{verbatim}
{ 
  %\set Staff.autoBeaming = ##f % turns off all autobeaming 
  \partCombine 
  { 
    \autoBeamOff % applies to split up stems 
    \repeat unfold 4 a'16 
    %\autoBeamOff % applies to combined up stems 
    \repeat unfold 4 a'8 
    \repeat unfold 4 a'16 
  } 
  { 
    \autoBeamOff % applies to down stems 
    \repeat unfold 4 f'8 
    \repeat unfold 8 f'16 |
  } 
}
\end{verbatim}
See also

Notation Reference: [Manual beams], page 96, [Setting automatic beam behavior], page 87.
Installed Files: scm/auto-beam.scm.
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings

The properties of a beam are determined at the start of its construction and any additional beam-property changes that occur before the beam has been completed will not take effect until the next, new beam starts.

Setting automatic beam behavior

When automatic beaming is enabled, the placement of automatic beams is determined by three context properties: baseMoment, beatStructure, and beamExceptions. The default values of these variables may be overridden as described below, or alternatively the default values themselves may be changed as explained in [Time signature], page 66.

If a beamExceptions rule is defined for the time signature in force, that rule alone is used to determine the beam placement; the values of baseMoment and beatStructure are ignored.

If no beamExceptions rule is defined for the time signature in force, the beam placement is determined by the values of baseMoment and beatStructure.

Beaming based on baseMoment and beatStructure

By default, beamExceptions rules are defined for most common time signatures, so the beamExceptions rules must be disabled if automatic beaming is to be based on baseMoment and beatStructure. The beamExceptions rules are disabled by

\set Timing.beamExceptions = #'()

When beamExceptions is set to #'(), either due to an explicit setting or because no beamExceptions rules are defined internally for the time signature in force, the ending points for beams are on beats as specified by the context properties baseMoment and beatStructure. beatStructure is a scheme list that defines the length of each beat in the measure in units of baseMoment. By default, baseMoment is one over the denominator of the time signature. By default, each unit of length baseMoment is a single beat.

Note that there are separate beatStructure and baseMoment values for each time signature. Changes to these variables apply only to the time signature that is currently in force, hence those changes must be placed after the \time command which starts a new time signature section, not before it. New values given to a particular time signature are retained and reinstated whenever that time signature is re-established.

\relative c' { 
\time 5/16 
c16"default" c c c c | 
% beamExceptions are unlikely to be defined for 5/16 time, 
% but let's disable them anyway to be sure 
\set Timing.beamExceptions = #'() 
\set Timing.beatStructure = 2,3 
c16"(2+3)" c c c c |
Beam setting changes can be limited to specific contexts. If no setting is included in a lower-level context, the setting of the enclosing context will apply.
When multiple voices are used the Staff context must be specified if the beaming is to be applied to all voices in the staff:

\time 7/8
% rhythm 3-1-1-2
% Change applied to Voice by default -- does not work correctly
% Because of autogenerated voices, all beating will
% be at baseMoment (1 . 8)
\set beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \relative {f'4. f8 f f f} >>

% Works correctly with context Staff specified
\set Staff.beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \relative {f'4. f8 f f f} >>

The value of baseMoment can be adjusted to change the beaming behavior, if desired. When this is done, the value of beatStructure must be set to be compatible with the new value of baseMoment.

\time 5/8
% No need to disable beamExceptions
% as they are not defined for 5/8 time
\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = 7,3
\repeat unfold 10 { a'16 }

baseMoment is a moment; a unit of musical duration. A quantity of type moment is created by the scheme function ly:make-moment. For more information about this function, see [Time administration], page 121.

By default baseMoment is set to one over the denominator of the time signature. Any exceptions to this default can be found in scm/time-signature-settings.scm.

**Beaming based on beamExceptions**

Special autobeaming rules (other than ending a beam on a beat) are defined in the beamExceptions property.

The value for beamExceptions, a somewhat complex Scheme data structure, is easiest generated with the \beamExceptions function. This function is given one or more manually beamed measure-length rhythmic patterns (measures have to be separated by a bar check | since the function has no other way to discern the measure length). Here is a simple example:

\relative c' { 
\time 3/16
\set Timing.beatStructure = 2,1
\set Timing.beamExceptions =
When the time signature is changed, default values of `Timing.baseMoment`, `Timing.beatStructure`, and `Timing.beamExceptions` are set. Setting the time signature will reset the automatic beaming settings for the `Timing` context to the default behavior.

```latex
\relative a' {
  \time 6/8
  \repeat unfold 6 { a8 }
  % group (4 + 2)
  \set Timing.beatStructure = 4,2
  \repeat unfold 6 { a8 }
  % go back to default behavior
  \time 6/8
  \repeat unfold 6 { a8 }
}
```

The default automatic beaming settings for a time signature are determined in `scm/time-signature-settings.scm`. Changing the default automatic beaming settings for a time signature is described in [Time signature], page 66.

Many automatic beaming settings for a time signature contain an entry for `beamExceptions`. For example, 4/4 time tries to beam the measure in two if there are only eighth notes. The `beamExceptions` rule can override the `beatStructure` setting if `beamExceptions` is not reset.

```latex
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = 3,3,2
% This won't beam (3 3 2) because of beamExceptions
\repeat unfold 8 {c'8} |
% This will beam (3 3 2) because we clear beamExceptions
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c'8}
```
In a similar fashion, eighth notes in 3/4 time are beamed as a full measure by default. To beam eighth notes in 3/4 time on the beat, reset \texttt{beamExceptions}.

\begin{verbatim}
\time 3/4
% by default we beam in (6) due to beamExceptions
\repeat unfold 6 \{a'8\} |
% This will beam (1 1 1) due to default baseMoment and beatStructure
\set Timing.beamExceptions = #'()
\repeat unfold 6 \{a'8\}
\end{verbatim}

In engraving from the Romantic and Classical periods, beams often begin midway through the measure in 3/4 time, but modern practice is to avoid the false impression of 6/8 time (see Gould, p. 153). Similar situations arise in 3/8 time. This behavior is controlled by the context property \texttt{beamHalfMeasure}, which has effect only in time signatures with 3 in the numerator:

\begin{verbatim}
\relative a' {
\time 3/4
r4. a8 a a |
\set Timing.beamHalfMeasure = ##f
r4. a8 a a |
}
\end{verbatim}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{beams}
\caption{Example of beaming in \texttt{3/4} time.}
\end{figure}

\section*{How automatic beaming works}

When automatic beaming is enabled, the placement of automatic beams is determined by the context properties \texttt{baseMoment}, \texttt{beatStructure}, and \texttt{beamExceptions}.

The following rules, in order of priority, apply when determining the appearance of beams:

- If a manual beam is specified with $[\ldots]$ set the beam as specified, otherwise
- If a beam-ending rule is defined in \texttt{beamExceptions} for the beam-type, use it to determine the valid places where beams may end, otherwise
- If a beam-ending rule is defined in \texttt{beamExceptions} for a longer beam-type, use it to determine the valid places where beams may end, otherwise
- Use the values of \texttt{baseMoment} and \texttt{beatStructure} to determine the ends of the beats in the measure, and end beams at the end of beats.

In the rules above, the \texttt{beam-type} is the duration of the shortest note in the beamed group.

The default beaming rules can be found in \texttt{scm/time-signature-settings.scm}.

\section*{Selected Snippets}

\subsection*{Subdividing beams}

The beams of consecutive 16th (or shorter) notes are, by default, not subdivided. That is, the three (or more) beams stretch unbroken over entire groups of notes. This behavior can be modified to subdivide the beams into sub-groups by setting the property \texttt{subdivideBeams}. When set, multiple beams will be subdivided at intervals defined by the current value of \texttt{baseMoment} by reducing the multiple beams to the number of beams that indicates the metric value of the
subdivision. If the group following the division is shorter than the current metric value (usually because the beam is incomplete) the number of beams reflects the longest possible subdivision group. However, if there is only one note left after the division this restriction isn’t applied. Note that baseMoment defaults to one over the denominator of the current time signature if not set explicitly. It must be set to a fraction giving the duration of the beam sub-group using the ly:make-moment function, as shown in this snippet. Also, when baseMoment is changed, beatStructure should also be changed to match the new baseMoment:

\relative c' { 
  c32[ c c c c c c c] 
  \set subdivideBeams = ##t 
  c32[ c c c c c c c] 

  \set baseMoment = #(ly:make-moment 1/8) 
  \set beatStructure = 2,2,2,2 
  c32[ c c c c c c c] 

  \set baseMoment = #(ly:make-moment 1/16) 
  \set beatStructure = 4,4,4,4 
  c32[ c c c c c c c] 

  \set baseMoment = #(ly:make-moment 1/32) 
  \set beatStructure = 2,2,2,2 
  c32[ c c c c c] r32 

  \set baseMoment = #(ly:make-moment 1/32) 
  \set beatStructure = 2,2,2,2 
  c32[ c c c c] r16, r2 
} 

Strict beat beaming

Beamlets can be set to point in the direction of the beat to which they belong. The first beam avoids sticking out flags (the default); the second beam strictly follows the beat.

\relative c' { 
  \time 6/8 
  a8. a16 a a 
  \set strictBeatBeaming = ##t 
  a8. a16 a a 
}
Chapter 1: Musical notation

Conducting signs measure grouping signs

Beat grouping within a measure is controlled by the context property `beatStructure`. Values of `beatStructure` are established for many time signatures in `scm/time-signature-settings.scm`. Values of `beatStructure` can be changed or set with \set. Alternatively, \time can be used to both set the time signature and establish the beat structure. For this, you specify the internal grouping of beats in a measure as a list of numbers (in Scheme syntax) before the time signature.

\time applies to the Timing context, so it will not reset values of `beatStructure` or `baseMoment` that are set in other lower-level contexts, such as Voice.

If the Measure_grouping_engraver is included in one of the display contexts, measure grouping signs will be created. Such signs ease reading rhythmically complex modern music. In the example, the 9/8 measure is grouped in two different patterns using the two different methods, while the 5/8 measure is grouped according to the default setting in `scm/time-signature-settings.scm`:

\score {
  \new Voice \relative c'' {\time 9/8
    g8 g d d g g a( bes g) | \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) | \time 4,5 9/8
    g8 g d d g g a( bes g) | \time 5/8
    a4. g4 |
  }
}
\layout {
  \context {\Staff
    \consists "Measure_grouping_engraver"
  }
}

Beam endings in Score context

Beam-ending rules specified in the Score context apply to all staves, but can be modified at both Staff and Voice levels:

\relative c'' {\time 5/4
% Set default beaming for all staves
\set Score.baseMoment = #(ly:make-moment 1/8)
\set Score.beatStructure = 3,4,3
<<
  \new Staff {
    c8 c c c c c c c c
  }

  \new Staff {
    \set Staff.beatStructure = 6,4
    c8 c c c c c c c c
  }

  \new Staff {
    \% Inherit beaming from Score context
    <<
      {
        \voiceOne
        c8 c c c c c c c c
      }
    \% Modify beaming for this voice only
    \new Voice {
      \voiceTwo
      \set Voice.beatStructure = 6,4
      a8 a a a a a a a a
    }
    >>
  }
>>

See also
Notation Reference: [Time signature], page 66.

Installed Files: scm/time-signature-settings.scm.

Snippets: Section “Rhythms” in Snippets.

Known issues and warnings

If a score ends while an automatic beam has not been ended and is still accepting notes, this last beam will not be typeset at all. The same holds for polyphonic voices, entered with \<< \ldots \ldots >>. If a polyphonic voice ends while an automatic beam is still accepting notes, it is not typeset. The workaround for these problems is to manually beam the last beam in the voice or score.

By default, the Timing translator is aliased to the Score context. This means that setting the time signature in one staff will affect the beaming of the other staves as well. Thus, a time signature setting in a later staff will reset custom beaming that was set in an earlier staff. One way to avoid this problem is to set the time signature in only one staff.

\begin{verbatim}
<<
\new Staff {
  \time 3/4
  \set Timing.baseMoment = #(ly:make-moment 1/8)
  \set Timing.beatStructure = 1,5
  \set Timing.beamExceptions = #'()
  \repeat unfold 6 { a'8 }
}
\new Staff {
  \repeat unfold 6 { a'8 }
}
>>
\end{verbatim}

The default beam settings for the time signature can also be changed, so that the desired beaming will always be used. Changes in automatic beaming settings for a time signature are described in [Time signature], page 66.

\begin{verbatim}
<<
\new Staff {
  \overrideTimeSignatureSettings
    3/4 % timeSignatureFraction
    1/8 % baseMomentFraction
    1,5 % beatStructure
    #'() % beamExceptions
  \time 3/4
  \repeat unfold 6 { a'8 }
}
\new Staff {
  \time 3/4
  \repeat unfold 6 { a'8 }
}
>>
\end{verbatim}
Manual beams

In some cases it may be necessary to override the automatic beaming algorithm. For example, the autobeamer will not put beams over rests or bar lines, and in choral scores the beaming is often set to follow the meter of the lyrics rather than the notes. Such beams can be specified manually by marking the begin and end point with \[ and \].

\relative { r4 r8[ g' a r] r g[ | a] r }

Beaming direction can be set manually using direction indicators:

\relative { c'8^[ d e] c,_[ d e f g] }

Individual notes may be marked with \noBeam to prevent them from being beamed:

\relative { \time 2/4 c''8 c \noBeam c c }

Grace note beams and normal note beams can occur simultaneously. Unbeamed grace notes are not put into normal note beams.

\relative { c''4 d8[ \grace { e32 d c d } e8] e[ e \grace { f16 } e8 e] }

Even more strict manual control with the beams can be achieved by setting the properties stemLeftBeamCount and stemRightBeamCount. They specify the number of beams to draw on the left and right side, respectively, of the next note. If either property is set, its value will be
used only once, and then it is erased. In this example, the last f is printed with only one beam on the left side, i.e., the eighth-note beam of the group as a whole.

\relative a' {
  a8[ r16 f g a]
  a8[ r16
  \set stemLeftBeamCount = #2
  \set stemRightBeamCount = #1
  f16
  \set stemLeftBeamCount = #1
  g16 a]
}

\score {%
  \new RhythmicStaff {
    % Example 1
    \set stemLeftBeamCount = #0
    c16[]
    r8.
  }
  % Example 2
  \new RhythmicStaff {
    % Example 3
    r8.
    \set stemRightBeamCount = #0
    16[]
  }
}

Predefined commands
\noBeam.

Selected Snippets

Flat flags and beam nibs

Flat flags on lone notes and beam nibs at the ends of beamed figures are both possible with a combination of \set stemLeftBeamCount, \set stemRightBeamCount and paired [] beam indicators.

For right-pointing flat flags on lone notes, use paired [] beam indicators and set \set stemLeftBeamCount to zero (see Example 1).

For left-pointing flat flags, set \set stemRightBeamCount instead (Example 2).

For right-pointing nibs at the end of a run of beamed notes, set \set stemRightBeamCount to a positive value. And for left-pointing nibs at the start of a run of beamed notes, set \set stemLeftBeamCount instead (Example 3).

Sometimes it may make sense for a lone note surrounded by rests to carry both a left- and right-pointing flat flag. Do this with paired [] beam indicators alone (Example 4).

(Note that \set stemLeftBeamCount is always equivalent to \once \set. In other words, the beam count settings are not “sticky”, so the pair of flat flags attached to the lone 16[] in the last example have nothing to do with the \set two notes prior.)

\score {%
  % Example 4
  % Example 1
  \new RhythmicStaff {
    % Example 2
    \set stemLeftBeamCount = #0
    c16[]
    r8.
  }
  % Example 3
  \new RhythmicStaff {
    % Example 4
    r8.
    \set stemRightBeamCount = #0
    16[]
  }
  % Example 3
}
Feathered beams

Feathered beams are used to indicate that a small group of notes should be played at an increasing (or decreasing) tempo, without changing the overall tempo of the piece. The extent of the feathered beam must be indicated manually using [ and ], and the beam feathering is turned on by specifying a direction to the Beam property grow-direction.

If the placement of the notes and the sound in the MIDI output is to reflect the ritardando or accelerando indicated by the feathered beam the notes must be grouped as a music expression delimited by braces and preceded by a featherDurations command which specifies the ratio between the durations of the first and last notes in the group.
The square brackets show the extent of the beam and the braces show which notes are to have their durations modified. Normally these would delimit the same group of notes, but this is not required: the two commands are independent.

In the following example the eight 16th notes occupy exactly the same time as a half note, but the first note is one half as long as the last one, with the intermediate notes gradually lengthening. The first four 32nd notes gradually speed up, while the last four 32nd notes are at a constant tempo.

\relative c' {
  \override Beam.grow-direction = #LEFT
  \featherDurations #(ly:make-moment 2/1)
  { c16[ c c c c c c c] }
  \override Beam.grow-direction = #RIGHT
  \featherDurations #(ly:make-moment 2/3)
  { c32[ d e f] }
  \% revert to non-feathered beams
  \override Beam.grow-direction = #'()
  { g32[ a b c] }
}

\relative { e'4 d c2 \bar "|." }

The spacing in the printed output represents the note durations only approximately, but the MIDI output is exact.

**Predefined commands**
\featherDurations.

**See also**
Snippets: Section “Rhythms” in Snippets.

**Known issues and warnings**
The \featherDurations command only works with very short music snippets, and when numbers in the fraction are small.

**1.2.5 Bars**

**Bar lines**
Bar lines delimit measures, and are also used to indicate repeats. Normally, simple bar lines are automatically inserted into the printed output at places based on the current time signature.

The simple bar lines inserted automatically can be changed to other types with the \bar command. For example, a closing double bar line is usually placed at the end of a piece:

\relative { e'4 d c2 \bar "|" }

It is not invalid if the final note in a measure does not end on the automatically entered bar line: the note is assumed to carry over into the next measure. But if a long sequence of such
carry-over measures appears the music can appear compressed or even flowing off the page. This is because automatic line breaks happen only at the end of complete measures, i.e., where all notes end before the end of a measure.

\begin{quote}
\begin{center}
\textbf{Note:} An incorrect duration can cause line breaks to be inhibited, leading to a line of highly compressed music or music that flows off the page.
\end{center}
\end{quote}

Line breaks are also permitted at manually inserted bar lines even within incomplete measures. To allow a line break without printing a bar line, use the following:

\begin{verbatim}
\bar ""
\end{verbatim}

This will insert an invisible bar line and allow (but not force) a line break to occur at this point. The bar number counter is not increased. To force a line break see Section 4.3.1 [Line breaking], page 560.

This and other special bar lines may be inserted manually at any point. When they coincide with the end of a measure they replace the simple bar line which would have been inserted there automatically. When they do not coincide with the end of a measure the specified bar line is inserted at that point in the printed output.

Note that manual bar lines are purely visual. They do not affect any of the properties that a normal bar line would affect, such as measure numbers, accidentals, line breaks, etc. They do not affect the calculation and placement of subsequent automatic bar lines. When a manual bar line is placed where a normal bar line already exists, the effects of the original bar line are not altered.

Two types of simple bar lines and five types of double bar lines are available for manual insertion:

\begin{verbatim}
\relative {
  f'1 \bar "|"
  f1 \bar "."
  g1 \bar "||"
  a1 \bar ".|
  b1 \bar ".."
  c1 \bar "|.|"
  d1 \bar "."
  e1
}
\end{verbatim}

together with dotted and dashed bar lines:

\begin{verbatim}
\relative {
  f'1 \bar ";"
  g1 \bar "!"
  a1
}
\end{verbatim}
and nine types of repeat bar lines:

\relative {
    f'1 \bar ".|:" 
    g1 \bar ":..:" 
    a1 \bar ":||:" 
    b1 \bar ":\.:" 
    c1 \bar ":.|.:" 
    d1 \bar ":[|:" 
    e1 \bar ":[][|:" 
    f1 \bar ":]\[|:" 
    g1 \bar ":.|." 
    a1 
}

Additionally, a bar line can be printed as a simple tick:

\relative {
    f'1 \bar "'|" g'1 
}

However, as such ticks are typically used in Gregorian chant, it is preferable to use \divisiMinima there instead, described in the section [Divisiones], page 459, in Gregorian chant.

LilyPond supports kievan notation and provides a special kievan bar line:

\relative {
    f'1 \bar "k" 
}

Further details of this notation are explained in Section 2.9.5 [Typesetting Kievian square notation], page 468.

For in-line segno signs, there are three types of bar lines which differ in their behavior at line breaks:

\relative c' { 
    c4 c c c 
    \bar "S" 
    c4 c c c \break 
    \bar "S" 
    c4 c c c 
    \bar "S-|" 
    c4 c c c \break 
    \bar "S-|" 
    c4 c c c 
    \bar "S-S" 
    c4 c c c \break 
}
Although the bar line types signifying repeats may be inserted manually they do not in themselves cause LilyPond to recognize a repeated section. Such repeated sections are better entered using the various repeat commands (see Section 1.4 [Repeats], page 151), which automatically print the appropriate bar lines.

In addition, you can specify ".|:-||", which is equivalent to "..:" except at line breaks, where it gives a double bar line at the end of the line and a start repeat at the beginning of the next line.

For combinations of repeats with the segno sign, there are six different variations:
Additionally there is an \inStaffSegno command which creates a segno bar line in conjunction with an appropriate repeat bar line when used with a \repeat volta command, see [Normal repeats], page 151.

New bar line types can be defined with \defineBarLine:

\defineBarLine bartype #'(end begin span)

In addition to bartype (the character string that will then be used to refer to that new bar line), it takes three values: the first two determine the bar line’s appearance when it occurs at a line break, in which case the first and second given glyphs are printed respectively at the end of the system and at the beginning of the next one. The third given glyph is only relevant in multi-staff systems (see [Grouping staves], page 193), where it is used as span bar, printed between staves.

The \defineBarLine variables can include the ‘empty’ string "", which is equivalent to an invisible bar line being printed. Or they can be set to #f which prints no bar line at all.

After the definition, the new bar line can be used by \bar bartype.

There are currently ten bar line elements available:

\defineBarLine ":" #'("" ":" ":")
\defineBarLine ":" #'("" ":" "")
\defineBarLine ":" #'("" ":" ""
\defineBarLine "[" #'("[ "" "")
\defineBarLine "]" #'("] "" "")
\defineBarLine "[" #'("" "")
\defineBarLine "]" #'("] "" "")

\new Staff {
  s1 \bar "|"
  s1 \bar "."
  s1 \bar "!"
  s1 \bar ";"
  s1 \bar ":"
  s1 \bar "k"
  s1 \bar "S"
  s1 \bar "="
  s1 \bar "["
  s1 \bar "]"
  s1 \bar ""
}

\new StaffGroup <<
  \new Staff \relative c' { 
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }

The "=" bar line provides the double span bar line, used in combination with the segno sign. Using it as a standalone double thin bar line is not recommended; in most cases, \bar "||" is preferred.

The "-" sign starts annotations to bar lines which are useful to distinguish those with identical appearance but different behavior at line breaks and/or different span bars. The part following the "-" sign is not used for building up the bar line.

\defineBarLine "||-dashedSpan" #'("|| "" "!")

\new StaffGroup <<
  \new Staff \relative c' { 
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }
Furthermore, the space character " " serves as a placeholder for defining span bars correctly aligned to the main bar lines:

\defineBarLine ":.|.-wrong" #'("." "" ".")
\defineBarLine ":.|.-right" #'("." "" ".")

\new StaffGroup <<
\new Staff \relative c'' {
  c1 \bar ":.|.-wrong"
  c1 \bar ":.|.-right"
  c1
}
\new Staff \relative c'' {
  c1
  c1
  c1
}
>>

New bar line types defined using \defineBarLine may even, in turn, be used in a second bar line definition. Such ‘nested’ definitions make it possible to use customized glyphs in places where it would not be otherwise possible, such as system ends:

\defineBarLine "||-dashEverywhere" #'("!!" "!!" "!!")
\defineBarLine "||-advancedDashSpan" #'("||-dashEverywhere" "" "!!")

\new StaffGroup <<
\new Staff \relative c'' {
  c1 \bar "||"
  c1 \bar "||-advancedDashSpan"
  c1 \bar "||-advancedDashSpan"
If additional elements are needed, LilyPond provides a simple way to define them. For more information on modifying or adding bar lines, see file `scm/bar-line.scm`.

In scores with many staves, a `\bar` command in one staff is automatically applied to all staves. The resulting bar lines are connected between different staves of a `StaffGroup`, `PianoStaff`, or `GrandStaff`.

```lilypond
\new StaffGroup <<
  \new Staff \relative { 
    e'4 d 
    \bar "||" 
    f4 e 
  }
  \new Staff \relative { \clef bass c'4 g e g }
>>
\new Staff \relative { \clef bass c'2 c2 }
```

The command `\bar bartype` is a shortcut for `\set Timing.whichBar = bartype`. A bar line is created whenever the `whichBar` property is set.

The default bar type used for automatically inserted bar lines is "|". This may be changed at any time with `\set Timing.defaultBarType = bartype`.

**See also**

Notation Reference: Section 4.3.1 [Line breaking], page 560, Section 1.4 [Repeats], page 151, [Grouping staves], page 193.
Bar numbers

Bar numbers are typeset by default at the start of every line except the first line. The number itself is stored in the \texttt{currentBarNumber} property, which is normally updated automatically for every measure. It may also be set manually:

\begin{verbatim}
\relative c' {
    c1 c c c
    \break
    \set Score.currentBarNumber = #50
    c1 c c c
}
\end{verbatim}

Bar numbers can be typeset at regular intervals instead of just at the beginning of every line. To do this the default behavior must be overridden to permit bar numbers to be printed at places other than the start of a line. This is controlled by the \texttt{break-visibility} property of \texttt{BarNumber}. This takes three values which may be set to \texttt{#t} or \texttt{#f} to specify whether the corresponding bar number is visible or not. The order of the three values is end of line visible, middle of line visible, beginning of line visible. In the following example bar numbers are printed at all possible places:

\begin{verbatim}
\relative c' {
    \override Score.BarNumber.break-visibility = ##(#t #t #t)
    \set Score.currentBarNumber = #11
    % Permit first bar number to be printed
    \bar ""
    c1 | c | c | c |
    \break
    c1 | c | c | c |
}
\end{verbatim}
Selected Snippets

Printing the bar number for the first measure

By default, the first bar number in a score is suppressed if it is less than or equal to ‘1’. By setting `barNumberVisibility` to `all-bar-numbers-visible`, any bar number can be printed for the first measure and all subsequent measures. Note that an empty bar line must be inserted before the first note for this to work.

```latex
\layout {
  indent = 0
  ragged-right = ##t
}
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```

Printing bar numbers at regular intervals

Bar numbers can be printed at regular intervals by setting the property `barNumberVisibility`. Here the bar numbers are printed every two measures except at the end of the line.

```latex
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```

---

Chapter 1: Musical notation 108
Printing bar numbers with changing regular intervals

The bar number interval can be changed by changing the context function \{set-bar-number-visibility\}.

\relative c' {
    \override Score.BarNumber.break-visibility = #end-of-line-invisible
    \context Score \applyContext #(set-bar-number-visibility 4)
    \repeat unfold 10 c'1
    \context Score \applyContext #(set-bar-number-visibility 2)
    \repeat unfold 10 c
}

Printing bar numbers inside boxes or circles

Bar numbers can also be printed inside boxes or circles.

\relative c' {
    % Prevent bar numbers at the end of a line and permit them elsewhere
    \override Score.BarNumber.break-visibility = #end-of-line-invisible
    \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

    % Increase the size of the bar number by 2
    \override Score.BarNumber.font-size = #2

    % Draw a box round the following bar number(s)
    \override Score.BarNumber.stencil
        = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
    \repeat unfold 5 { c1 }

    % Draw a circle round the following bar number(s)
    \override Score.BarNumber.stencil
        = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
    \repeat unfold 4 { c1 } \bar "|."
}
Alternative bar numbering

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

\relative c'\{
\set Score.alternativeNumberingStyle = #'numbers
\repeat volta 3 { c4 d e f | }
\alternative {
{ c4 d e f | c2 d \break }
{ f4 g a b | f4 g a b | f2 a | \break }
{ c4 d e f | c2 d }
}
c1 \break
\set Score.alternativeNumberingStyle = #'numbers-with-letters
\repeat volta 3 { c,4 d e f | }
\alternative {
{ c4 d e f | c2 d \break }
{ f4 g a b | f4 g a b | f2 a | \break }
{ c4 d e f | c2 d }
}
c1\}

1.  
2.  
3.  
5.  
6b.  
6c.
Aligning bar numbers

Bar numbers by default are right-aligned to their parent object. This is usually the left edge of a line or, if numbers are printed within a line, the left hand side of a bar line. The numbers may also be positioned directly over the bar line or left-aligned to the bar line.

\relative c' {
    \set Score.currentBarNumber = #111
    \override Score.BarNumber.break-visibility = #all-visible
    \override Score.BarNumber.font-size = #2
    \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
    c1 | c1
    \override Score.BarNumber.self-alignment-X = #CENTER
    c1 | c1
    \override Score.BarNumber.self-alignment-X = #LEFT
    c1 | c1
}

Removing bar numbers from a score

Bar numbers can be removed entirely by removing the Bar_number_engraver from the Score context.

\layout {
    \context {
        \Score
        \omit BarNumber
        \or:
        \remove "Bar_number_engraver"
    }
}

\relative c'' {
    c4 c c c \break
    c4 c c c
}

Removing bar numbers from a score

Bar numbers can be removed entirely by removing the Bar_number_engraver from the Score context.

\layout {
    \context {
        \Score
        \omit BarNumber
        \or:
        \remove "Bar_number_engraver"
    }
}

\relative c'' {
    c4 c c c \break
    c4 c c c
}
See also

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “BarNumber” in Internals Reference, Section “Bar_number_engraver” in Internals Reference.

Known issues and warnings

Bar numbers may collide with the top of the StaffGroup bracket, if there is one. To solve this, the padding property of BarNumber can be used to position the number correctly. See Section “StaffGroup” in Internals Reference and Section “BarNumber” in Internals Reference for more.

Bar and bar number checks

Bar checks help detect errors in the entered durations. A bar check may be entered using the bar symbol, |, at any place where a bar line is expected to fall. If bar check lines are encountered at other places, a list of warnings is printed in the log file, showing the line numbers and lines in which the bar checks failed. In the next example, the second bar check will signal an error.

\time 3/4 c2 e4 | g2 |

An incorrect duration can result in a completely garbled score, especially if the score is polyphonic, so a good place to start correcting input is by scanning for failed bar checks and incorrect durations.

If successive bar checks are off by the same musical interval, only the first warning message is displayed. This allows the warning to focus on the source of the timing error.

Bar checks can also be inserted in lyrics:

\lyricmode {
\time 2/4
Twin -- k1e | Twin -- k1e |
}

Note that bar check marks in lyrics are evaluated at the musical moment when the syllable following the check mark is processed. If the lyrics are associated with the notes of a voice which has a rest at the beginning of a bar, then no syllable can be located at the start of that bar and a warning will be issued if a bar check mark is placed in the lyrics at that position.

It is also possible to redefine the action taken when a bar check or pipe symbol, |, is encountered in the input, so that it does something other than a bar check. This is done by assigning a music expression to "|". In the following example | is set to insert a double bar line wherever it appears in the input, rather than checking for end of bar.

"|" = \bar "||"
{
 c'2  c' |
 c'2  c' |
 c'2  c' |
 c'2  c' |
}

When copying large pieces of music, it can be helpful to check that the LilyPond bar number corresponds to the original that you are entering from. This can be checked with \barNumberCheck, for example,
\barNumberCheck #123
will print a warning if the currentBarNumber is not 123 when it is processed.

**See also**

Snippets: Section “Rhythms” in Snippets.

### Rehearsal marks

To print a rehearsal mark, use the \mark command.

\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}

![Rehearsal mark example](noteheads.s0)

The mark is incremented automatically if you use \mark \default, but you can also use an integer argument to set the mark manually. The value to use is stored in the property rehearsalMark.

\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}

![Rehearsal mark example](noteheads.s0)

The letter 'I' is skipped in accordance with engraving traditions. If you wish to include the letter 'I', then use one of the following commands, depending on which style of rehearsal mark you want (letters only, letters in a hollow box, or letters in a hollow circle).

\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet

\relative c'' {
  \set Score.markFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}

![Rehearsal mark example](noteheads.s0)
The style is defined by the property markFormatter. It is a function taking the current mark (an integer) and the current context as argument. It should return a markup object. In the following example, markFormatter is set to a pre-defined procedure. After a few measures, it is set to a procedure that produces a boxed number.

\relative c' {
  \set Score.markFormatter = \format-mark-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = \format-mark-box-numbers
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-letters
  c1
}

\relative c' {
  \set Score.markFormatter = \format-mark-box-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-letters
  c1
}

\relative c' {
  \set Score.markFormatter = \format-mark-box-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = \format-mark-circle-letters
  c1
}

The file scm/translation-functions.scm contains the definitions of format-mark-letters (the default format), format-mark-box-letters, format-mark-numbers and format-mark-box-numbers. These can be used as inspiration for other formatting functions.

You may use format-mark-barnumbers, format-mark-box-barnumbers, and format-mark-circle-barnumbers to get bar numbers instead of incremented numbers or letters.

Other styles of rehearsal mark can be specified manually:
\mark "A1"

Note that Score.markFormatter does not affect marks specified in this manner. However, it is possible to apply a \markup to the string.
\mark \markup { \box A1 }

Music glyphs (such as the segno sign) may be printed inside a \mark
\relative c' {
  c1 \mark \markup { \musicglyph "scripts.segno" }
  c1 \mark \markup { \musicglyph "scripts.coda" }
  c1 \mark \markup { \musicglyph "scripts.ufermata" }
  c1
}

See Section A.8 [The Emmentaler font], page 683, for a list of symbols which may be printed with \musicglyph.

For common tweaks to the positioning of rehearsal marks, see Section 1.8.2 [Formatting text], page 252. For more precise control, see break-alignable-interface in Section 5.5.1 [Aligning objects], page 650.
The file `scm/translation-functions.scm` contains the definitions of `format-mark-numbers` and `format-mark-letters`. They can be used as inspiration for other formatting functions.

See also

Notation Reference: Section A.8 [The Emmentaler font], page 683, Section 1.8.2 [Formatting text], page 252, Section 5.5.1 [Aligning objects], page 650.

Installed Files: `scm/translation-functions.scm`.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “MarkEvent” in Internals Reference, Section “Mark_engraver” in Internals Reference, Section “RehearsalMark” in Internals Reference.

1.2.6 Special rhythmic concerns

Grace notes

Grace notes are musical ornaments, printed in a smaller font, that take up no additional logical time in a measure.

```
\relative {
  \grace { b16 c16 } a2
}
```

There are three other types of grace notes possible; the *acciaccatura* – an unmeasured grace note indicated by a slurred note with a slashed stem – and the *appoggiatura*, which takes a fixed fraction of the main note it is attached to and prints without the slash. It is also possible to write a grace note with a slashed stem, like the *acciaccatura* but without the slur, so as to place it between notes that are slurred themselves, using the `\slashedGrace` function.

```
\relative {
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
    \slashedGrace b8 a2)
}
```

The placement of grace notes is synchronized between different staves. In the following example, there are two sixteenth grace notes for every eighth grace note

```
<<
  \new Staff \relative { e''2 \grace { c16 d e f } e2 }
  \new Staff \relative { c''2 \grace { g8 b } c2 }
>>
```
If you want to end a note with a grace, use the `\afterGrace` command. It takes two arguments: the main note, and the grace notes following the main note.

\[ \textbackslash relative \{ c'1 \textbackslash afterGrace d1 \{ c16[ d] \} c1 \} \]

This will place the grace notes after the start of the main note. The point of time where the grace notes are placed is a given fraction of the main note’s duration. The default setting of `afterGraceFraction = 3/4` may be redefined at top level. Individual `\afterGrace` commands may have the fraction specified right after the command itself instead.

The following example shows the results from setting with the default space, setting it at 15/16, and finally at 1/2 of the main note.

\[
<< \\
\textbackslash new \textbackslash Staff \textbackslash relative \{ \\
 c''1 \textbackslash afterGrace d1 \{ c16[ d] \} c1 \\
\} \\
\textbackslash new \textbackslash Staff \textbackslash relative \{ \\
 c''1 \textbackslash afterGrace 15/16 d1 \{ c16[ d] \} c1 \\
\} \\
\textbackslash new \textbackslash Staff \textbackslash relative \{ \\
 c''1 \textbackslash afterGrace 1/2 d1 \{ c16[ d] \} c1 \\
\} \\
>>
\]

The effect of `\afterGrace` can also be achieved using spacers. The following example places the grace note after a space lasting 7/8 of the main note.

\[ \textbackslash new \textbackslash Voice \textbackslash relative \{ \\
<< \\
\{ d''1\textbackslash trill\_\} \\
\{ s2 s4. \textbackslash grace \{ c16 d \} \} \\
>> \\
c1 \} \]
A \texttt{\textbackslash grace} music expression will introduce special typesetting settings, for example, to produce smaller type, and set directions. Hence, when introducing layout tweaks to override the special settings, they should be placed inside the grace expression. The overrides should also be reverted inside the grace expression. Here, the grace note’s default stem direction is overridden and then reverted.

\begin{verbatim}
\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
}
\end{verbatim}

Selected Snippets

\textit{Using grace note slashes with normal heads}

The slash through the stem found in acciaccaturas can be applied in other situations.

\begin{verbatim}
\relative c' {  
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}
\end{verbatim}

\textit{Tweaking grace layout within music}

The layout of grace expressions can be changed throughout the music using the functions \texttt{\textbackslash add-grace-property} and \texttt{\textbackslash remove-grace-property}. The following example undefines the Stem direction for this grace, so that stems do not always point up, and changes the default note heads to crosses.

\begin{verbatim}
\relative c' {  
  \new Staff {  
    $(\texttt{\textbackslash remove-grace-property 'Voice 'Stem 'direction})
    $(\texttt{\textbackslash add-grace-property 'Voice 'NoteHead 'style 'cross})
  \new Voice {  
    \acciaccatura { f16 } g4  
    \grace { d16 e } f4  
    \appoggiatura { f,32 g a } e2
  }
}
\end{verbatim}
Redefining grace note global defaults

The global defaults for grace notes are stored in the identifiers `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` and `stopAppoggiaturaMusic`, which are defined in the file `ly/grace-init.ly`. By redefining them other effects may be obtained.

```latex
startAcciaccaturaMusic = {
  <<
    \override Flag.stroke-style = #"grace"
    \slurDashed
  >>
}
stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <<
}

\relative c'' {
  \acciaccatura d8 c1
}
```

Positioning grace notes with floating space

Setting the property `'strict-grace-spacing` makes the musical columns for grace notes 'floating', i.e., decoupled from the non-grace notes: first the normal notes are spaced, then the (musical columns of the) graces are put left of the musical columns for the main notes.

```latex
\relative c'' {
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```
See also


Notation Reference: [Scaling durations], page 54, [Manual beams], page 96.

Installed Files: ly/grace-init.ly.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “GraceMusic” in Internals Reference, Section “Grace_beam_engraver” in Internals Reference, Section “Grace_auto_beam_engraver” in Internals Reference, Section “Grace_engraver” in Internals Reference, Section “Grace_spacing_engraver” in Internals Reference.

Known issues and warnings

A multi-note beamed acciaccatura is printed without a slash, and looks exactly the same as a multi-note beamed appoggiatura.

Grace note synchronization can also lead to surprises. Staff notation, such as key signatures, bar lines, etc., are also synchronized. Take care when you mix staves with grace notes and staves without, for example,

```
<<
\new Staff \relative { e''4 \bar ":.\:" \grace c16 d2. }
\new Staff \relative { c''4 \bar ":.\:" d2. }
>>
```

This can be remedied by inserting grace skips of the corresponding durations in the other staves. For the above example

```
<<
\new Staff \relative { e''4 \bar ":.\:" \grace c16 d2. }
\new Staff \relative { c''4 \bar ":.\:" \grace s16 d2. }
>>
```
Please make sure that you use the `\grace` command for the spacer part, even if the visual part uses `\acciaccatura` or `\appoggiatura` because otherwise an ugly slur fragment will be printed, connecting the invisible grace note with the following note.

Grace sections should only be used within sequential music expressions. Nesting or juxtaposing grace sections is not supported, and might produce crashes or other errors.

Each grace note in MIDI output has a length of 1/4 of its actual duration. If the combined length of the grace notes is greater than the length of the preceding note a “Going back in MIDI time” error will be generated. Either make the grace notes shorter in duration, for example:

\begin{verbatim}
c'8 \acciaccatura { c'8[ d' e' f' g' ] }
\end{verbatim}

becomes:

\begin{verbatim}
c'8 \acciaccatura { c'16[ d' e' f' g' ] }
\end{verbatim}

Or explicitly change the musical duration:

\begin{verbatim}
c'8 \acciaccatura \scaleDurations 1/2 { c'8[ d' e' f' g' ] } }
\end{verbatim}

See [Scaling durations], page 54.

\section*{Aligning to cadenzas}

In an orchestral context, cadenzas present a special problem: when constructing a score that includes a measured cadenza or other solo passage, all other instruments should skip just as many notes as the length of the cadenza, otherwise they will start too soon or too late.

One solution to this problem is to use the functions `mmrest-of-length` and `skip-of-length`. These Scheme functions take a defined piece of music as an argument and generate a multi-measure rest or `\skip` exactly as long as the piece.

\begin{verbatim}
MyCadenza = \relative {
  c'4 d8 e f g g4
  f2 g4 g
}
\end{verbatim}

\begin{verbatim}
\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
    c'1
  }
>>
\end{verbatim}
See also


Snippets: Section “Rhythms” in Snippets.

Time administration

Time is administered by the Timing_translator, which by default is to be found in the Score context. An alias, Timing, is added to the context in which the Timing_translator is placed. To ensure that the Timing alias is available, you may need to explicitly instantiate the containing context (such as Voice or Staff).

The following properties of Timing are used to keep track of timing within the score.

currentBarNumber

The current measure number. For an example showing the use of this property see [Bar numbers], page 107.

measureLength

The length of the measures in the current time signature. For a 4/4 time this is 1, and for 6/8 it is 3/4. Its value determines when bar lines are inserted and how automatic beams should be generated.

measurePosition

The point within the measure where we currently are. This quantity is reset by subtracting measureLength whenever measureLength is reached or exceeded. When that happens, currentBarNumber is incremented.

timing

If set to true, the above variables are updated for every time step. When set to false, the engraver stays in the current measure indefinitely.

Timing can be changed by setting any of these variables explicitly. In the next example, the default 4/4 time signature is printed, but measureLength is set to 5/4. At 4/8 through the third measure, the measurePosition is advanced by 1/8 to 5/8, shortening that bar by 1/8. The next bar line then falls at 9/8 rather than 5/4.

\new Voice \relative {    \set Timing.measureLength = #(ly:make-moment 5/4)    c'1 c4 |    c1 c4 |    c4 c    \set Timing.measurePosition = #(ly:make-moment 5/8)    b4 b b8 |    c4 c1 |  }

As the example illustrates, ly:make-moment n/m constructs a duration of n/m of a whole note. For example, ly:make-moment 1/8 is an eighth note duration and ly:make-moment 7/16 is the duration of seven sixteenths notes.

See also

Notation Reference: [Bar numbers], page 107, [Unmetered music], page 75.

Snippets: Section “Rhythms” in Snippets.
1.3 Expressive marks

1.3.1 Expressive marks attached to notes

This section explains how to create expressive marks that are attached to notes: articulations, ornamentations, and dynamics. Methods to create new dynamic markings are also discussed.

Articulations and ornamentations

A variety of symbols that denote articulations, ornamentations, and other performance indications can be attached to a note using this syntax:

\relative { 
    c'\'4\staccato c\mordent b2\turn 
    c1\fermata 
}

This section lists various expressive marks that can be created in a score.
Some of these articulations have shorthands for easier entry. Shorthands are appended to the note name, and their syntax consists of a dash - followed by a symbol signifying the articulation. Predefined shorthands exist for marcato, stopped, tenuto, staccatissimo, accent, staccato, and portato. Their corresponding output appears as follows:

\relative {  
c''4-= c-= c-- c-!
c4-> c-. c2--  
}

\override Script.color = #red
\override MultiMeasureRestScript.color = #blue
\relative a'2 \fermata \fermata
R1 \fermata

The rules for the default placement of articulations are defined in $\texttt{scm/script.scm}$. Articulations and ornamentations may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 634.

Articulations are Script objects. Their properties are described more fully in Section “Script” in Internals Reference.

Articulations can be attached to rests and multi-measure rests as well as notes. Attaching an articulation to a multi-measure rest creates a MultiMeasureRestScript object.

\relative c'' { c1-+  
\override MultiMeasureRestScript.color = #red  
a'2 \fermata \fermata  
R1 \fermata  
}

In addition to articulations, text and markups can be attached to notes. See [Text scripts], page 245.

For more information about the ordering of Scripts and TextScripts that are attached to the notes, see Section “Placement of objects” in Learning Manual.

**Selected Snippets**

*Modifying default values for articulation shorthand notation*

The shorthands are defined in $\texttt{ly/script-init.ly}$, where the variables dashHat, dashPlus, dashDash, dashBang, dashLarger, dashDot, and dashUnderscore are assigned default values. The default values for the shorthands can be modified. For example, to associate the $-$+ (dashPlus) shorthand with the trill symbol instead of the default + symbol, assign the value trill to the variable dashPlus:

\relative c'' { c1-+  
  
dashPlus = "trill"  
\relative c'' { c1-+  

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{example.png}
\caption{Example of a multi-measure rest with articulations.}
\end{figure}
Controlling the vertical ordering of scripts

The vertical ordering of scripts is controlled with the ‘script-priority’ property. The lower this number, the closer it will be put to the note. In this example, the TextScript (the sharp symbol) first has the lowest priority, so it is put lowest in the first example. In the second, the prall trill (the Script) has the lowest, so it is on the inside. When two objects have the same priority, the order in which they are entered determines which one comes first.

\relative c''' {
\once \override TextScript.script-priority = #-100
a2\prall\markup { \sharp }
\once \override Script.script-priority = #-100
a2\prall\markup { \sharp }
}

Creating a delayed turn

Creating a delayed turn, where the lower note of the turn uses the accidental, requires several overrides. The outside-staff-priority property must be set to \#f, as otherwise this would take precedence over the avoid-slur property. Changing the fractions 2/3 and 1/3 adjusts the horizontal position.

\relative c'' {
c2*2/3 ( s2*1/3\turn d4) r
<<
{ c4.( d8) }
{ s4 s\turn }
>>
\transpose c d \relative c'' <<
{ c4.( d8) }
{
 s4
 \once \set suggestAccidentals = ##t
 \once \override AccidentalSuggestion.outside-staff-priority = ##f
 \once \override AccidentalSuggestion.avoid-slur = '#inside
 \once \override AccidentalSuggestion.font-size = -3
 \once \override AccidentalSuggestion.script-priority = -1
 \single \hideNotes
 b8-\turn \noBeam
 s8
}
}
See also


Notation Reference: [Text scripts], page 245, Section 5.4.2 [Direction and placement], page 634, Section A.14 [List of articulations], page 768, [Trills], page 148.

Installed Files: scm/script.scm.

Snippets: Section “Expressive marks” in Snippets.

Internals Reference: Section “Script” in Internals Reference, Section “TextScript” in Internals Reference.

Dynamics

Absolute dynamic marks are specified using a command after a note, such as \( c4\ff \). The available dynamic marks are \( \pppp, \pppp, \ppp, \pp, \p, \mp, \mf, \f, \ff, \fff, \ffff, \fpp, \sp, \sff, \sp, \spf, \spz, \spz, \p, \mp, \mf, \f, \ff, \fff, \ffff \). Dynamic marks may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 634.

\[
\begin{align*}
&\text{mf} & \text{ff} \\
&\pppp & \mp \\
&\rfz & \spp \\
&\sff & \\
&\sp \\
&\sp \\
&\sp \\
&\sp \\
\end{align*}
\]

A crescendo mark is started with \( \(< \) \) and terminated with \( \! \), an absolute dynamic, or an additional crescendo or decrescendo mark. A decrescendo mark is started with \( \( > \) \) and is also terminated with \( \! \), an absolute dynamic, or another crescendo or decrescendo mark. \( \text{cr} \) and \( \text{decr} \) may be used instead of \(< \) and \( > \); \( \text{endcr} \) and \( \text{enddecr} \) maybe used instead of \( \! \) to end a crescendo or decrescendo mark, respectively. Hairpins are engraved by default using this notation.

\[
\begin{align*}
&\text{mf} & \text{ff} \\
&\pppp & \mp \\
&\rfz & \spp \\
&\sff & \\
&\sp \\
&\sp \\
&\sp \\
&\sp \\
\end{align*}
\]

A hairpin that is terminated with \( \! \) will end at the right edge of the note that has the \( \! \) assigned to it. In the case where it is terminated with the start of another crescendo or
**decrescendo** mark, it will end at the centre of the note that has the next `<` or `>` assigned to it. The next hairpin will then start at the right edge of the same note instead of the usual left edge had it been terminated with `!` before. A hairpin ending on a downbeat will stop at the preceding bar line.

\[
\begin{align*}
\text{\texttt{relative \{} & c'\'1\< | c4 a c\< a | c4 a c! a\< | c4 a c a! \\
\texttt{\}}}
\end{align*}
\]

Hairpins that are terminated with absolute dynamic marks instead of `!` will also be engraved in a similar way. However, the length of the absolute dynamic itself can alter where the preceding hairpin ends.

\[
\begin{align*}
\text{\texttt{relative \{} & c'\'1\< | c4 a c\! a | c1\< | c4 a c\! fff a \\
\texttt{\}}}
\end{align*}
\]

Spacer rests are needed to engrave multiple marks on one note. This is particularly useful when adding a crescendo and decrescendo to the same note:

\[
\begin{align*}
\text{\texttt{relative \{} & c'\'4\< c\! d\> e\! \\
\texttt{<< f1 \{} & s4 s4\< s4\> s4\!
\texttt{\} >>}
\end{align*}
\]

The \texttt{espressivo} command can be used to indicate a crescendo and decrescendo on the same note. However, be warned that this is implemented as an articulation, not a dynamic.

\[
\begin{align*}
\text{\texttt{relative \{} & c'\'2 b4 a \\
\texttt{g1\texttt{\textbackslash espressivo}}
\end{align*}
\]

Textual crescendo marks begin with \texttt{\textbackslash cresc}. Textual decrescendos begin with \texttt{\textbackslash decresc} or \texttt{\textbackslash dim}. Extender lines are engraved as required.

\[
\begin{align*}
\text{\texttt{relative \{} & g'8\texttt{\textbackslash cresc a b c b c d e\textbackslash mf |}
\texttt{f8\textbackslash decresc e d c e\textbackslash d c b |}
\end{align*}
\]
Textual marks for dynamic changes can also replace hairpins:
\relative c' {\crescTextCresc
c4\< d e f\! | \dimTextDecresc
g4\> e d c\! | \dimTextDecr
e4\> d c b\! | \dimTextDim
d4\> c b a\! | \crescHairpin \dimHairpin
c4\< d\! e\> d\! |}

To create new absolute dynamic marks or text that should be aligned with dynamics, see [New dynamic marks], page 131.

Vertical positioning of dynamics is handled by Section “DynamicLineSpanner” in Internals Reference.

A Dynamics context is available to engrave dynamics on their own horizontal line. Use spacer rests to indicate timing. (Notes in a Dynamics context will also take up musical time, but will not be engraved.) The Dynamics context can usefully contain some other items such as text scripts, text spanners, and piano pedal marks.
Predefined commands
\dynamicUp, \dynamicDown, \dynamicNeutral, \crescTextCresc, \dimTextDim, \dimTextDecr, \dimTextDecresc, \crescHairpin, \dimHairpin.

Selected Snippets

Setting hairpin behavior at bar lines
If the note which ends a hairpin falls on a downbeat, the hairpin stops at the bar line immediately preceding. This behavior can be controlled by overriding the 'to-barline' property.
\relative c'' {
   e4\< e2.
   e1!\!
   \override Hairpin.to-barline = ##f
   e4\< e2.
   e1!\!
}

Setting the minimum length of hairpins
If hairpins are too short, they can be lengthened by modifying the minimum-length property of the Hairpin object.
\relative c'' {
   c4\< c\! d\> e\!
   << f1 { s4 s\< s\> s\! } >>
   \override Hairpin.minimum-length = #5
   << f1 { s4 s\< s\> s\! } >>
}

Moving the ends of hairpins
The ends of hairpins may be offset by setting the shorten-pair property of the Hairpin object. Positive values move endpoints to the right, negative to the left. Unlike the minimum-length property, this property only affects the appearance of the hairpin; it does not adjust horizontal spacing (including the position of bounding dynamics). This method is thus suitable for fine-tuning a hairpin within its allotted space.
{
   c'1~\<
   c'2~ c'!\!
   \once \override Hairpin.shorten-pair = '#(2 . 2)
   c'1~\<
   c'2~ c'!\!
Printing hairpins using *al niente* notation

Hairpin dynamics may be printed with a circled tip ("al niente" notation) by setting the circled-tip property of the Hairpin object to \#t.

\relative c'' {
\override Hairpin.circled-tip = ##t
  c\> c\!
  c\> c\!
}\n
Printing hairpins in various styles

Hairpin dynamics may be created in a variety of styles.

\relative c'' {
\override Hairpin.stencil = #flared-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
\override Hairpin.stencil = #constante-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
\override Hairpin.stencil = #flared-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
}\n
\relative c'' {
\override Hairpin.stencil = #flared-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
\override Hairpin.stencil = #constante-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
}\n
\relative c'' {
\override Hairpin.stencil = #flared-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
\override Hairpin.stencil = #constante-hairpin
  a\> a a \fff
  a\> a a \fff
  a\> a a \fff
}\n
Vertically aligned dynamics and textscripts

All DynamicLineSpanner objects (hairpins and dynamic texts) are placed with their reference line at least \texttt{staff-padding} from the staff, unless other notation forces them to be farther. Setting \texttt{staff-padding} to a sufficiently large value aligns the dynamics.

The same idea, together with \texttt{\textLengthOn}, is used to align the text scripts along their baseline.

\begin{verbatim}
music = \relative c' {  
a'2\p b\f  
e4\p f\f\> g, b\p  
c2\markup { \huge gorgeous } c\markup { \huge fantastic }  
}  
\end{verbatim}

\begin{verbatim}
{  
\music  
\break  
\override DynamicLineSpanner.staff-padding = #3  
\textLengthOn  
\override TextScript.staff-padding = #1  
\music  
}
\end{verbatim}

\begin{verbatim}
music = \relative c' {  
\relative c'' {  
\Override DynamicTextSpanner.style = #'none  
\crescTextCresc  
c1\< | d | b | c\!  
}  
}
\end{verbatim}

Hiding the extender line for text dynamics

Text style dynamic changes (such as cresc. and dim.) are printed with a dashed line showing their extent. This line can be suppressed in the following way:

\begin{verbatim}
\relative c' {  
\override DynamicTextSpanner.style = #'none  
\crescTextCresc  
c1\< | d | b | c\!  
}
\end{verbatim}

Changing text and spanner styles for text dynamics

The text used for crescendos and decrescendos can be changed by modifying the context properties \texttt{crescendoText} and \texttt{decrescendoText}. 
The style of the spanner line can be changed by modifying the 'style property of DynamicTextSpanner. The default value is 'dashed-line, and other possible values include 'line, 'dotted-line and 'none.

\relative c' {
    \set crescendoText = \markup { \italic { cresc. poco } }
    \set crescendoSpanner = #'text
    \override DynamicTextSpanner.style = #'dotted-line
    a2\text < a
    a2 a
    a2 a
    a2 a\mf
}

See also


Notation Reference: Section 5.4.2 [Direction and placement], page 634, [New dynamic marks], page 131, Section 3.5.9 [Enhancing MIDI output], page 542, Section 3.5.4 [Controlling MIDI dynamics], page 533.

Snippets: Section “Expressive marks” in Snippets.


New dynamic marks

The easiest way to create dynamic indications is to use \markup objects.

moltoF = \markup { molto \dynamic f }

\relative {
    <d' e>16_\moltoF <d e>
    <d e>2..
}

New dynamic marks

In markup mode, editorial dynamics (within parentheses or square brackets) can be created. The syntax for markup mode is described in Section 1.8.2 [Formatting text], page 252.

roundF = \markup {
    \center-align \concat { \bold { \italic ( \dynamic f \bold { \italic ) } } }
}

boxF = \markup { \bracket { \dynamic f } }
Simple, centered dynamic marks are easily created with the \texttt{make-dynamic-script} function.

\begin{verbatim}
sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfzp c}
\end{verbatim}

In general, \texttt{make-dynamic-script} takes any markup object as its argument. The dynamic font only contains the characters \texttt{f,m,p,r,s} and \texttt{z}, so if a dynamic mark that includes plain text or punctuation symbols is desired, markup commands that reverts font family and font encoding to normal text should be used, for example \texttt{\normal-text}. The interest of using \texttt{make-dynamic-script} instead of an ordinary markup is ensuring the vertical alignment of markup objects and hairpins that are attached to the same note head.

\begin{verbatim}
roundF = \markup { \center-align \concat {
  \normal-text { \textbf{\textit{ (} } }
  \textbf{f}
  \normal-text { \textbf{\textit{ )} } } } }
boxF = \markup { \textbf{f} }
mfEspress = \markup { \textbf{mf espress.} }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative {
  c'4\roundFdynamic<d e f
g,1\boxFdynamic>
g1
g'1\mfEspressDynamic
g1}
\end{verbatim}
The Scheme form of markup mode may be used instead. Its syntax is explained in Section “Markup construction in Scheme” in Extending.

\begin{verbatim}
\relative { 
  \moltoF = #(make-dynamic-script 
      (markup #:normal-text "molto" 
      #:dynamic "f"))

  \relative {
    \<d' e>16 \<d e> 
    \<d e>2..\moltoF 
  }

  \relative {
    \<d' e>16 \<d e> 
    \<d e>2..\moltoF <d e>1
  }
}
\end{verbatim}

Font settings in markup mode are described in [Selecting font and font size], page 254.

See also

Notation Reference: Section 1.8.2 [Formatting text], page 252, [Selecting font and font size], page 254, Section 3.5.9 [Enhancing MIDI output], page 542, Section 3.5.4 [Controlling MIDI dynamics], page 533.

Extending LilyPond: Section “Markup construction in Scheme” in Extending.

Snippets: Section “Expressive marks” in Snippets.

1.3.2 Expressive marks as curves

This section explains how to create various expressive marks that are curved: normal slurs, phrasing slurs, breath marks, falls, and doits.

Slurs

Slurs are entered using parentheses:

\begin{verbatim}
\relative {
  f'4( g a) a8 b(
}
\end{verbatim}

Note: In polyphonic music, a slur must be terminated in the same voice it began.
Slurs may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 634.

Simultaneous or overlapping slurs require special attention. Most occurrences of outer slurs actually indicate phrasing, and phrasing slurs may overlap a regular slur, see [Phrasing slurs], page 136. When multiple regular slurs are needed in a single Voice, matching slur starts and ends need to be labelled by preceding them with \= followed by an identifying key (a symbol or non-negative integer).

\fixed c' {
  \<c c\>2\>2 <c e\>1 ( a\>2 ) >
}

Slurs can be solid, dotted, or dashed. Solid is the default slur style:

\relative {
  c'4( e g2 )
  \slurDashed
  g4( e c2 )
  \slurDotted
  c4( e g2 )
  \slurSolid
  g4( e c2 )
}

Slurs can also be made half-dashed (the first half dashed, the second half solid) or half-solid (the first half solid, the second half dashed):

\relative {
  c'4( e g2 )
  \slurHalfDashed
  g4( e c2 )
  \slurHalfSolid
  c4( e g2 )
  \slurSolid
  g4( e c2 )
}
Custom dash patterns for slurs can be defined:
\relative {
  c'4( e g2)
  \slurDashPattern #0.7 #0.75
  g4( e c2)
  \slurDashPattern #0.5 #2.0
  c4( e g2)
  \slurSolid
  g4( e c2)
}\n
Predefined commands
\slurUp, \slurDown, \slurNeutral, \slurDashed, \slurDotted, \slurHalfDashed, \slurHalfSolid, \slurDashPattern, \slurSolid.

Selected Snippets

Using double slurs for legato chords
Some composers write two slurs when they want legato chords. This can be achieved by setting doubleSlurs.
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}\n
Positioning text markups inside slurs
Text markups need to have the outside-staff-priority property set to false in order to be printed inside slurs.
\relative c' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(\markup { \halign #-10 \natural } d4.) c8
}\n
Making slurs with complex dash structure
Slurs can be made with complex dash patterns by defining the `dash-definition` property. `dash-definition` is a list of `dash-elements`. A `dash-element` is a list of parameters defining the dash behavior for a segment of the slur.

The slur is defined in terms of the bezier parameter \( t \) which ranges from 0 at the left end of the slur to 1 at the right end of the slur. `dash-element` is a list \( (start-t\ stop-t\ dash-fraction\ dash-period) \). The region of the slur from \( start-t \) to \( stop-t \) will have a fraction `dash-fraction` of each `dash-period` black. `dash-period` is defined in terms of staff spaces. `dash-fraction` is set to 1 for a solid slur.

\[
\text{\texttt{\relative c' { \once \override Slur.dash-definition = #'((0 0.3 0.1 0.75) (0.3 0.6 1 1) (0.65 1.0 0.4 0.75)) c4( d e f) \once \override Slur.dash-definition = #'((0 0.25 1 1) (0.3 0.7 0.4 0.75) (0.75 1.0 1 1)) c4( d e f) }}}
\]

See also

Notation Reference: Section 5.4.2 [Direction and placement], page 634, [Phrasing slurs], page 136.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “Slur” in Internals Reference.

Phrasing slurs

Phrasing slurs (or phrasing marks) that indicate a musical sentence are written using the commands `\(` and `\)` respectively:

\[
\text{\texttt{\relative { c'4\( d( e) f( e2) d\) }}}
\]

Typographically, a phrasing slur behaves almost exactly like a normal slur. However, they are treated as different objects; a `\slurUp` will have no effect on a phrasing slur. Phrasing may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 634.
Simultaneous or overlapping phrasing slurs are entered using `\=` as with regular slurs, see [Slurs], page 133.

Phrasing slurs can be solid, dotted, or dashed. Solid is the default style for phrasing slurs:

```
\relative {  
c'4\{ e g2\}  
\phrasingSlurDashed  
g4\{ e c2\}  
\phrasingSlurDotted  
c4\{ e g2\}  
\phrasingSlurSolid  
g4\{ e c2\}  
}
```

Phrasing slurs can also be made half-dashed (the first half dashed, the second half solid) or half-solid (the first half solid, the second half dashed):

```
\relative {  
c'4\{ e g2\}  
\phrasingSlurHalfDashed  
g4\{ e c2\}  
\phrasingSlurHalfSolid  
c4\{ e g2\}  
\phrasingSlurSolid  
g4\{ e c2\}  
}
```

Custom dash patterns for phrasing slurs can be defined:

```
\relative {  
c'4\{ e g2\}  
\phrasingSlurDashPattern #0.7 #0.75  
g4\{ e c2\}  
\phrasingSlurDashPattern #0.5 #2.0  
c4\{ e g2\}  
\phrasingSlurSolid  
g4\{ e c2\}  
}
```

Dash pattern definitions for phrasing slurs have the same structure as dash pattern definitions for slurs. For more information about complex dash patterns, see the snippets under [Slurs], page 133.
Predefined commands
\phrasingSlurUp, \phrasingSlurDown, \phrasingSlurNeutral, \phrasingSlurDashed, \phrasingSlurDotted, \phrasingSlurHalfDashed, \phrasingSlurHalfSolid, \phrasingSlurDashPattern, \phrasingSlurSolid.

See also
Notation Reference: Section 5.4.2 [Direction and placement], page 634, [Slurs], page 133.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “PhrasingSlur” in Internals Reference.

Breath marks
Breath marks are entered using \breathe:

\{ c''2 \breathe d''4 \}

Unlike other expressive marks, a breath mark is not associated with the preceding note but is a separate music event. So all the expressive marks which are attached to the preceding note, any square brackets indicating manual beams, and any brackets indicating slurs and phrasing slurs must be placed before \breathe.

A breath mark will end an automatic beam; to override this behavior, see [Manual beams], page 96.

\relative { c''8 \breathe d e f g2 }

Musical indicators for breath marks in ancient notation, divisiones, are supported. For details, see [Divisiones], page 459.

Selected Snippets
Changing the breath mark symbol
The glyph of the breath mark can be tuned by overriding the text property of the BreathingSign layout object with any markup text.

\relative c' { c2 \override BreathingSign.text = \markup { \musicglyph "scripts.rvarcomma" } \breathe d2 }
Using a tick as the breath mark symbol

Vocal and wind music frequently uses a tick mark as a breathing sign. This indicates a breath that subtracts a little time from the previous note rather than causing a short pause, which is indicated by the comma breath mark. The mark can be moved up a little to take it away from the stave.

\relative c'' { 
    c2
    \breathe
    d2
    \override BreathingSign.Y-offset = #2.6
    \override BreathingSign.text = 
        \markup { \musicglyph "scripts.tickmark" }
    c2
    \breathe
    d2
}

[Staff Image]

Inserting a caesura

Caesura marks can be created by overriding the ‘text property of the BreathingSign object. A curved caesura mark is also available.

\relative c'' { 
    \override BreathingSign.text = \markup { 
        \musicglyph "scripts.caesura.straight"
    }
    c8 e4. \breathe g8. e16 c4

    \override BreathingSign.text = \markup { 
        \musicglyph "scripts.caesura.curved"
    }
    g8 e'4. \breathe g8. e16 c4
}

[Staff Image]

See also

Notation Reference: [Divisiones], page 459.
Snippets: Section “Expressive marks” in Snippets.
Falls and doits

Falls and doits can be added to notes using the `\bendAfter` command. The direction of the fall or doit is indicated with a plus or minus (up or down). The number indicates the pitch interval that the fall or doit will extend beyond the main note.

\[
\text{relative c'} \{ \\
\text{c2}\text{\bendAfter #+4} \\
\text{c2}\text{\bendAfter #\text{-}4} \\
\text{c2}\text{\bendAfter #+6.5} \\
\text{c2}\text{\bendAfter #\text{-}6.5} \\
\text{c2}\text{\bendAfter #+8} \\
\text{c2}\text{\bendAfter #\text{-}8}
\}
\]

Selected Snippets

Adjusting the shape of falls and doits

The shortest-duration-space property may be tweaked to adjust the shape of falls and doits.

\[
\text{relative c'} \{ \\
\text{\override Score.SpacingSpanner.shortest-duration-space} = \#4.0 \\
\text{c2}\text{\-\bendAfter #5} \\
\text{c2}\text{\-\bendAfter #\text{-}4.75} \\
\text{c2}\text{\-\bendAfter #8.5} \\
\text{c2}\text{\-\bendAfter #\text{-}6}
\}
\]

See also

Snippets: Section “Expressive marks” in Snippets.

1.3.3 Expressive marks as lines

This section explains how to create various expressive marks that follow a linear path: glissandos, arpeggios, and trills.

Glissando

A glissando is created by appending `\glissando` to a note:

\[
\text{relative} \{ \\
\text{g'2}\text{\glissando g'} \\
\text{c2}\text{\glissando c,} \\
\text{\afterGrace f,}\text{\glissando f'16}
\}
\]
A glissando can connect notes across staves:
\new PianoStaff <<
\new Staff = "right" {
   e''2\glissando
   \change Staff = "left"
   a,,4\glissando
   \change Staff = "right"
   b''8 r |
}
\new Staff = "left" {
   \clef bass
   s1
}
>>

A glissando can connect notes in chords. If anything other than a direct one-to-one pairing of
the notes in the two chords is required the connections between the notes are defined by setting
\glissandoMap, where the notes of a chord are assumed to be numbered from zero in the order
in which they appear in the input .ly file.
\relative {
   <c' e>1\glissando g' |
   <c, e>1\glissando |
   <g' b> |
   \break
   \set glissandoMap = #'((0 . 1) (1 . 0))
   <c, g'>1\glissando |
   <d a'> |
   \set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
   c1\glissando |
   <d f a> |
   \set glissandoMap = #'((2 . 0) (1 . 0) (0 . 1))
   <f d a'>1\glissando |
   <c c'> |
}
Different styles of glissandi can be created. For details, see Section 5.4.8 [Line styles], page 648.

**Selected Snippets**

*Contemporary glissando*

A contemporary glissando without a final note can be typeset using a hidden note and cadenza timing.

```latex
\relative c' ' {
    \time 3/4
    \override Glissando.style = #'zigzag
c4 c
\cadenzaOn
c4/glissando
\hideNotes
c,, 4
\unHideNotes
\cadenzaOff
\bar "|"
}
```

*Adding timing marks to long glissandi*

Skipped beats in very long glissandi are sometimes indicated by timing marks, often consisting of stems without noteheads. Such stems can also be used to carry intermediate expression markings.

If the stems do not align well with the glissando, they may need to be repositioned slightly.

```latex
\glissandoSkipOn = {
    \override NoteColumn.glissando-skip = ##t
    \hide NoteHead
    \override NoteHead.no-ledgers = ##t
}
\glissandoSkipOff = {
    \revert NoteColumn.glissando-skip
    \undo \hide NoteHead
    \revert NoteHead.no-ledgers
}
```

```latex
\relative c' ' {
    r8 f8/glissando
    \glissandoSkipOn
    f4 g a a8/noBeam
    \glissandoSkipOff
    a8

    r8 f8/glissando
    \glissandoSkipOn
```
Making glissandi breakable

Setting the `breakable` property to `#t` in combination with `after-line-breaking` allows a glissando to break if it occurs at a line break:

```latex
\glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
```

Extending glissandi across repeats

A glissando which extends into several `\alternative` blocks can be simulated by adding a hidden grace note with a glissando at the start of each `\alternative` block. The grace note
should be at the same pitch as the note which starts the initial glissando. This is implemented here with a music function which takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```music
repeatGliss = #(define-music-function (grace)
    (ly:pitch?)
    #% the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
        = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
    
\score {
    \relative c'' {
        \repeat volta 3 { c4 d e f\glissando }
        \alternative {
            { g2 d }
            { \repeatGliss f g2 e }
            { \repeatGliss f e2 d }
        }
    }
}

music = \relative c' {
    \voiceOne
    \repeat volta 2 {
        g a b c\glissando
    }
    \alternative {
        { d1 }
        { \repeatGliss c \once \omit StringNumber e1\2 }
    }
}

\score {
    \new StaffGroup <<
        \new Staff <<
            \context Voice { \clef "G_8" \music }
        >>
    \new TabStaff <<
        \context TabVoice { \clef "moderntab" \music }
        >>
        >>
    }
```
Chapter 1: Musical notation

See also
Notation Reference: Section 5.4.8 [Line styles], page 648.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “Glissando” in Internals Reference.

Known issues and warnings
Printing text over the line (such as gliss.) is not supported.

Arpeggio
An arpeggio on a chord (also known as a broken chord) is denoted by appending \arpeggio to the chord construct:

\relative { <c' e g c>1\arpeggio }

Different types of arpeggios may be written. \arpeggioNormal reverts to a normal arpeggio:

\relative {
<e g c>2\arpeggio
\arpeggioArrowUp
<e g c>2\arpeggio
\arpeggioArrowDown
<e g c>2\arpeggio
\arpeggioNormal
<e g c>2\arpeggio
}

These predefined commands internally modify the 'arpeggio-direction property; see their full definition in the ly/property-init.ly file.
Special *bracketed* arpeggio symbols can be created:

\relative {
  <c' e g c>2
  \arpeggioBracket <c e g c>2\arpeggio
  \arpeggioParenthesis <c e g c>2\arpeggio
  \arpeggioParenthesisDashed <c e g c>2\arpeggio
  \arpeggioNormal <c e g c>2\arpeggio
}

These predefined commands internally override the Arpeggio object’s ‘stencil’ property, and may also adapt its ‘X-extent’ (that is, the horizontal dimension it takes not to collide with other objects).

The dash properties of the parenthesis arpeggio are controlled with the ‘dash-definition’ property, which is described at [Slurs], page 133.

Arpeggios can be explicitly written out with ties. For more information, see [Ties], page 55.

Predefined commands
\arpeggio, \arpeggioArrowUp, \arpeggioArrowDown, \arpeggioNormal, \arpeggioBracket, \arpeggioParenthesis, \arpeggioParenthesisDashed.

Selected Snippets

*Creating cross-staff arpeggios in a piano staff*

In a PianoStaff, it is possible to let an arpeggio cross between the staves by setting the property PianoStaff.connectArpeggios.

\new PianoStaff \relative c' \relative c' <<
\set PianoStaff.connectArpeggios = ##t
\new Staff {
  <c e g c>4\arpeggio
  <g e g c>4\arpeggio
  <e g c e>4\arpeggio
  <c e g c>4\arpeggio
}
\new Staff {
  \clef bass
  \repeat unfold 4 {
    <c,,, e g c>4\arpeggio
  }
}
Creating cross-staff arpeggios in other contexts

Cross-staff arpeggios can be created in contexts other than GrandStaff, PianoStaff and StaffGroup if the Span_arpeggio_engraver is included in the Score context.

\score {
\new ChoirStaff {
  \set Score.connectArpeggios = ##t
<<
  \new Voice \relative c' {
    <c e>2\arpeggio
    <d f>2\arpeggio
    <c e>1\arpeggio
  }
  \new Voice \relative c {
    \clef bass
    <c g'>2\arpeggio
    <b g'>2\arpeggio
    <c g'>1\arpeggio
  }
  >>
}
\layout {
  \context {
    \Score
    \consists "Span_arpeggio_engraver"
  }
}
}

Creating arpeggios across notes in different voices

An arpeggio can be drawn across notes in different voices on the same staff if the Span_arpeggio_engraver is added to the Staff context:

\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' { 
  \set Staff.connectArpeggios = ##t 
  << 
    { <e' g>4\arpeggio <d f> <d f>2 } 
    \ 
    { <d, f>2\arpeggio <g b>2 } 
  >> 
}

\override Staff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
\override PianoStaff.Arpeggio.arpeggio-direction = #UP

See also

Notation Reference: [Slurs], page 133, [Ties], page 55.
Installed Files: ly/property-init.ly.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “Arpeggio” in Internals Reference, Section “Slur” in Internals Reference, Section “PianoStaff” in Internals Reference.

Known issues and warnings

Predefined commands such as \arpeggioArrowUp only apply to the current context, and thus will not affect arpeggios spanning several voices or staves. In such cases, these commands need to be used in a \context block within \layout, or in a \with block, as explained in Section 5.1.5 [Changing context default settings], page 607. Alternatively, rather than using predefined shortcuts, it may be advisable to directly override the relevant properties for the Arpeggio object in the appropriate context; for example:

\override Staff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
to print cross-voice arpeggio brackets at the Staff level, or
\override PianoStaff.Arpeggio.arpeggio-direction = #UP
to print cross-staff arrowed arpeggios (pointing upwards) in a PianoStaff context.

It is not possible to mix connected arpeggios and unconnected arpeggios in one PianoStaff at the same point in time.

The simple way of setting parenthesis-style arpeggio brackets does not work for cross-staff arpeggios; see [Cross-staff stems], page 345.

Trills

Short trills without an extender line are printed with \trill; see [Articulations and ornamentations], page 122.

Longer trills with an extender line are made with \startTrillSpan and \stopTrillSpan:

\relative { 
  d'\startTrillSpan 
  d1 
  c2\stopTrillSpan 
  r2 
}
A trill spanner crossing a line break will restart exactly above the first note on the new line.
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan
  r2
}

Consecutive trill spans will work without explicit \stopTrillSpan commands, since successive trill spanners will automatically become the right bound of the previous trill.
\relative {
  d''1\startTrillSpan
  d1
  b1\startTrillSpan
  d2\stopTrillSpan
  r2
}

Trills can also be combined with grace notes. The syntax of this construct and the method to precisely position the grace notes are described in \[Grace notes\], page 115.
\relative {
  d''1\afterGrace
  d1\startTrillSpan { c32[ d]\stopTrillSpan }
  c2 r2
}

Trills that require an auxiliary note with an explicit pitch can be typeset with the \pitchedTrill command. The first argument is the main note, and the second is the trilled note, printed as a stemless note head in parentheses.
\relative {
  \pitchedTrill
The Accidental of the first pitched trill in a measure is always printed, even for naturals.

\begin{verbatim}
{\key d \major \pitchedTrill d'2\startTrillSpan cis d\stopTrillSpan \pitchedTrill d2\startTrillSpan c d\stopTrillSpan \pitchedTrill d2\startTrillSpan e d\stopTrillSpan \pitchedTrill d2\startTrillSpan e d\stopTrillSpan \pitchedTrill eis''4\startTrillSpan fis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan cis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan fis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan fis! eis4\stopTrillSpan eis4\stopTrillSpan}
\end{verbatim}

Subsequent accidentals (of the same note in the same measure) will need to be added manually.

\begin{verbatim}
{\relative 
  {\pitchedTrill eis''4\startTrillSpan fis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan cis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan fis eis4\stopTrillSpan \pitchedTrill eis4\startTrillSpan fis! eis4\stopTrillSpan eis4\stopTrillSpan}
\end{verbatim}

Predefined commands
\startTrillSpan, \stopTrillSpan.
Repetition is a central concept in music, and multiple notations exist for repetitions. LilyPond supports the following kinds of repeats:

**volta** The repeated music is not written out but enclosed between repeat bar lines. If the repeat is at the beginning of a piece, a repeat bar line is only printed at the end of the repeat. Alternative endings (volte) are printed left to right with brackets. This is the standard notation for repeats with alternatives.

**unfold** The repeated music is fully written out, as many times as specified by `repeatcount`. This is useful when entering repetitious music.

**percent** These are beat or measure repeats. They look like single slashes or percent signs.

**tremolo** This is used to write tremolo beams.

### 1.4.1 Long repeats

This section discusses how to input long (usually multi-measure) repeats. The repeats can take two forms: repeats enclosed between repeat signs; or written-out repeats, used to input repetitious music. Repeat signs can also be controlled manually.

**Normal repeats**

The syntax for a normal repeat is

```
\repeat volta repeatcount musicexpr
```

where `musicexpr` is a music expression.
A single repeat without an alternate ending:
\begin{verbatim}
\relative {
    \repeat volta 2 { c''4 d e f } 
    c2 d 
    \repeat volta 2 { d4 e f g } 
}
\end{verbatim}

An ‘opening’ repeat mark is not, by default, printed in the first full measure. However it is possible to add one by using \( \bar{\cdot}: \) before the first note.
\begin{verbatim}
\relative {
    \repeat volta 2 { \bar{\cdot}: c''4 d e f } 
    c2 d 
    \repeat volta 2 { d4 e f g } 
}
\end{verbatim}

Alternative endings can be produced using \texttt{\alternative}. Each group of alternatives must be themselves, enclosed in a set of braces.
\begin{verbatim}
\repeat volta \repeatcount musicexpr \alternative {
    \{ musicexpr \}
}
\end{verbatim}

where \texttt{musicexpr} is a music expression.

If there are more repeats than there are alternate endings, the earliest repeats are given the first alternative.

A single repeat with one alternate ending:
\begin{verbatim}
\relative {
    \repeat volta 2 { c''4 d e f | } 
    \alternative {
        \{ c2 e | \}
        \{ f2 g | \}
    }
    c1
}
\end{verbatim}

Multiple repeats with one alternate ending:
\begin{verbatim}
\relative {
    \repeat volta 4 { c''4 d e f | } 
    \alternative {
\end{verbatim}
Multiple repeats with more than one alternate ending:
\relative {  
  \repeat volta 3 { c''4 d e f | }  
  \alternative {  
    { c2 e | }  
    { f2 g | }  
    { a2 g | }  
  }  
}

\relative {  
  \repeat volta 3 { c''4 d e f | }  
  \alternative {  
    { c2 e | }  
    { f2 g | }  
    { a2 g | }  
  }  
}

\relative {  
  \repeat volta 3 { c''4 d e f | }  
  \alternative {  
    { c2 e | }  
    { f2 g | }  
    { a2 g | }  
  }  
}

\relative {  
  \repeat volta 3 { c''4 d e f | }  
  \alternative {  
    { c2 e | }  
    { f2 g | }  
    { a2 g | }  
  }  
}

Note: If there are two or more alternatives, nothing should appear between the closing brace of one and the opening brace of the next in an \alternative block, otherwise you will not get the expected number of endings.

Note: If you include \relative inside a \repeat without explicitly instantiating the Voice context, extra (unwanted) staves will appear. See Section “An extra staff appears” in Application Usage.

If a repeat that has no alternate endings starts in the middle of a measure, it will usually end at a corresponding place in the middle of a later measure (so that the two ends add up to one complete measure). In this case the repeat signs are not ‘true’ bar lines so neither bar checks nor \partial commands should be placed there:

\relative {  
  \repeat volta 4 {  
    e4 |  
    c2 e |  
    g4 g g  
  }  
  \repeat volta 4 {  
    e4 |  
    c2 e |  
    g4 g g  
  }  
  \repeat volta 4 {  
    e4 |  
    c2 e |  
    g4 g g  
  }  
  \repeat volta 4 {  
    e4 |  
    c2 e |  
    g4 g g  
  }  
}
If a repeat that has no alternate endings starts with a partial measure, then the same principles apply, except that a \texttt{\partial} command is required at the start of the measure:

\begin{verbatim}
\texttt{\partial 4} \\
\texttt{\repeat volta 4 { \\
\texttt{\textbackslash e'4 |} \\
\texttt{\textbackslash c2 e |} \\
\texttt{g4 g g} \\
\texttt{g4 |} \\
\texttt{a2 a |} \\
\texttt{g1 |}
}}
\end{verbatim}

Ties may be added to a second ending:

\begin{verbatim}
\texttt{\relative { \\
\texttt{c''1} \\
\texttt{\repeat volta 2 { c4 d e f~ } } \\
\texttt{\alternative { \\
\texttt{\{ f2 d } \\
\texttt{\{ f2\texttt{\repeatTie f, } } \\
\texttt{\}} \\
\texttt{\}}}
}}
\end{verbatim}

The \texttt{\inStaffSegno} command can be used to generate a composite bar line incorporating the segno symbol with the appropriate repeat bar line when used with the \texttt{\repeat volta} command. The correct type of repeat bar line, viz. start repeat, end repeat or double repeat, is selected automatically. Note that the corresponding “D.S.” mark must be added manually.

Away from a repeat:

\begin{verbatim}
\texttt{\relative { \\
\texttt{e'1} \\
\texttt{\inStaffSegno} \\
\texttt{f2 g a b} \\
\texttt{c1_"D.S." \textbackslash bar "|."}
}}
\end{verbatim}

At the start of a repeat:

\begin{verbatim}
\texttt{\relative { \\
}}
\end{verbatim}
At the end of a repeat:
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }% start repeat
  \inStaffSegno % end repeat
  f2 g a b
  c1_"D.S." \bar "."
}

Between two repeats:
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }% start repeat
  \inStaffSegno % double repeat
  \repeat volta 2 {
    f2 g a b
  }% end repeat
  c1_"D.S." \bar "."
}

Alternative bar line symbols can be obtained by setting (in the Score context) the properties `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` or `doubleRepeatSegnoType` to the required bar line type. The alternative bar line types must be selected from the pre-defined types or types previously defined with the `\defineBarLine` command (see [Bar lines], page 99).
\defineBarLine ":|.|." ":.|." "S[" "]"
\defineBarLine ":|.|." ":.|." "S[" "]"
\relative {

Selected Snippets

Shortening volta brackets

By default, the volta brackets will be drawn over all of the alternative music, but it is possible to shorten them by setting \voltaSpannerDuration. In the next example, the bracket only lasts one measure, which is a duration of 3/4.

\relative c' { 
  \time 3/4
  c4 c c 
  \set Score.voltaSpannerDuration = #(ly:make-moment 3/4) 
  \repeat volta 5 { d4 d d } 
  \alternative { 
    { e4 e e 
    f4 f f 
    } 
    { g4 g g } 
  } 
}

Adding volta brackets to additional staves

The \Volta_engraver by default resides in the \Score context, and brackets for the repeat are thus normally only printed over the topmost staff. This can be adjusted by adding the \Volta_engraver to the \Staff context where the brackets should appear; see also the “Volta multi staff” snippet.

<< 
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } } 
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } } 
\new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' } 
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } } 
>>
Setting the double repeat default for volte

There are three different styles of double repeats for volte, that can be set using `doubleRepeatType`.

\[\begin{aligned}
\relative c'' { \\
\set Score.doubleRepeatType = #"::..:" \\
\repeat volta 1 { c1 } \\
\set Score.doubleRepeatType = #"::|.|:" \\
\repeat volta 1 { c1 } \\
\set Score.doubleRepeatType = #"::|.:" \\
\repeat volta 1 { c1 } \\
}\end{aligned}\]

Alternative bar numbering

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

\[\begin{aligned}
\relative c' { \\
\set Score.alternativeNumberingStyle = #'numbers \\
\repeat volta 3 { c4 d e f | } \\
\alternative { \\
\{ c4 d e f | c2 d \break \} \\
\{ f4 g a b | f4 g a b | f2 a | \break \} \\
\{ c4 d e f | c2 d \} \\
} \\
c1 \break \\
\set Score.alternativeNumberingStyle = #'numbers-with-letters \\
\repeat volta 3 { c,4 d e f | } \\
\alternative { \\
\{ c4 d e f | c2 d \break \} \\
\{ f4 g a b | f4 g a b | f2 a | \break \} \\
\{ c4 d e f | c2 d \} \\
} \\
c1 \\
}\end{aligned}\]
See also


Notation Reference: [Bar lines], page 99, Section 5.1.4 [Modifying context plug-ins], page 605, [Modifying ties and slurs], page 655, [Time administration], page 121.

Installed Files: ly/engraver-init.ly.

Snippets: Section “Repeats” in Snippets.


Known issues and warnings

Slurs that span from a \repeat block into an \alternative block will only work for the first alternative ending. The visual appearance of a continuing slur in other alternative blocks may be simulated with \repeatTie if the slur extends into only one note in the alternative block, although this method does not work in TabStaff. Other methods which may be tailored to indicate continuing slurs over several notes in alternative blocks, and which also work in TabStaff contexts, are shown in [Modifying ties and slurs], page 655.

Also, slurs cannot wrap around from the end of one alternative back to the beginning of the repeat.

Glissandi that span from a \repeat block into an \alternative block will only work for the first alternative ending. The visual appearance of a continuing glissando in other alternative blocks may be indicated by coding a glissando starting on a hidden grace note. For an example, see “Extending glissandi across repeats” under Selected Snippets in [Glissando], page 140.
If a repeat that begins with an incomplete measure has an `\alternative` block that contains modifications to the `measureLength` property, using `\unfoldRepeats` will result in wrongly-placed bar lines and bar check warnings.

A nested repeat like
```
\repeat ...
\repeat ...
\alternative
```
is ambiguous, since it is not clear to which `\repeat` the `\alternative` belongs. This ambiguity is resolved by always having the `\alternative` belong to the inner `\repeat`. For clarity, it is advisable to use braces in such situations.

### Manual repeat marks

| Note: These methods are only used for displaying unusual repeat constructs, and may produce unexpected behavior. In most cases, repeats should be created using the standard `\repeat` command or by printing the relevant bar lines. For more information, see [Bar lines], page 99. |

The property `repeatCommands` can be used to control the layout of repeats. Its value is a Scheme list of repeat commands.

**start-repeat**

Print a `.|:` bar line.
```
\relative {
  c'1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
c1
}
```

As per standard engraving practice, repeat signs are not printed at the beginning of a piece.

**end-repeat**

Print a `:|.` bar line:
```
\relative {
  c'1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
c1
}
```
(volta number) ... (volta #f)

Create a new volta with the specified number. The volta bracket must be explicitly terminated, or it will not be printed.

\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2"))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}

Multiple repeat commands may occur at the same point:

\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2, 5") end-repeat)
  g4 a g a
  c1
  \set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
  b1
  \set Score.repeatCommands = #'((volta #f))
}

Text can be included with the volta bracket. The text can be a number or numbers or markup text, see Section 1.8.2 [Formatting text], page 252. The simplest way to use markup text is to define the markup first, then include the markup in a Scheme list.

voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative {
  c''1
  \set Score.repeatCommands = #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
See also
Notation Reference: [Bar lines], page 99, Section 1.8.2 [Formatting text], page 252.
Snippets: Section “Repeats” in Snippets.

Written-out repeats
By using the unfold command, repeats can be used to simplify the writing out of repetitious music. The syntax is
\repeat unfold repeatcount musicexpr
wheremusicexpris a music expression and repeatcount is the number of times musicexpr is repeated.
\repeat unfold 2 { c''4 d e f }
c1

In some cases, especially in a relativecontext, the \repeat unfold function is not the same as writing out the music expression multiple times. E.g,
\repeat unfold 2 { a'4 b c }
is not equivalent to
a'4 b c | a'4 b c
Unfold repeats can be made with alternate endings.
\repeat unfold 2 { c''4 d e f }
\alternative {
    { c'2 g' }
    { c,2 b }
}
c1

If there are more repeats than there are alternate endings, the first alternative is applied multiple times until the remaining alternatives make up the total number of repeats.
\repeat unfold 4 { c''4 d e f }
\alternative {
    { c'2 g' }
    { c,2 b }
    { e2 d }
}
If there are more alternate endings than repeats then only the first alternatives are applied. The remaining alternatives will be ignored and not printed.

\relative {
  \repeat unfold 2 { c''4 d e f }
  \alternative {
    \{ c2 g' \}
    \{ c,2 b \}
    \{ e2 d \}
  }
  c1
}

It is also possible to nest multiple unfold functions (with or without alternate endings).

\relative {
  \repeat unfold 2 {
    \repeat unfold 2 { c''4 d e f }
    \alternative {
      \{ c2 g' \}
      \{ c,2 b \}
    }
  }
  c1
}

Chord constructs can be repeated by the chord repetition symbol q. See [Chord repetition], page 170.

Note: If you include \relative inside a \repeat without explicitly instantiating the Voice context, extra (unwanted) staves will appear. See Section “An extra staff appears” in Application Usage.

See also

Notation Reference: [Chord repetition], page 170.
Snippets: Section “Repeats” in Snippets.
1.4.2 Short repeats

This section discusses how to input short repeats. Short repeats can take two forms: slashes or percent signs to represent repeats of a single note, a single measure or two measures, and tremolos otherwise.

Percent repeats

Repeated short patterns are printed once, and the repeated pattern is replaced with a special sign.

The syntax is

\repeat percent number musicexpr

where musicexpr is a music expression.

Patterns that are shorter than one measure are replaced by slashes.

\relative c'' {
  \repeat percent 4 { c128 d e f }
  \repeat percent 4 { c64 d e f }
  \repeat percent 5 { c32 d e f }
  \repeat percent 4 { c16 d e f }
  \repeat percent 4 { c8 d }
  \repeat percent 4 { c4 }
  \repeat percent 2 { c2 }
}

Patterns of one or two measures are replaced by percent-like symbols.

\relative c'' {
  \repeat percent 2 { c4 d e f }
  \repeat percent 2 { c2 d }
  \repeat percent 2 { c1 }
}

\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}

\relative {
  \repeat percent 2 { c4 d e f }
  \repeat percent 2 { c2 d }
  \repeat percent 2 { c1 }
}

\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}
Patterns that are shorter than one measure but contain mixed durations use a double-percent symbol.

\relative {
    \repeat percent 4 { c''8. <d f>16 }
    \repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
}

Selected Snippets

*Percent repeat counter*

Measure repeats of more than two repeats can get a counter when the convenient property is switched, as shown in this example:

\relative c'' {
    \set countPercentRepeats = ##t
    \repeat percent 4 { c1 }
}

*Percent repeat count visibility*

Percent repeat counters can be shown at regular intervals by setting the context property `repeatCountVisibility`.

\relative c'' {
    \set countPercentRepeats = ##t
    \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
    \repeat percent 10 { c1 } \break
    \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
    \repeat percent 6 { c1 d1 }
}

*Isolated percent repeats*

Isolated percents can also be printed.

\makePercent =

#(define-music-function (note) (ly:music?)
    "Make a percent repeat the same length as NOTE."
    (make-music 'PercentEvent
Chapter 1: Musical notation

\[ \text{\texttt{\textbackslash length (ly:music-length note}\})} \]

\begin{verbatim}
\relative c' { 
  \makePercent s1 }
\end{verbatim}

See also


Snippets: Section “Repeats” in Snippets.

Internals Reference: Section “RepeatSlash” in Internals Reference, Section “Repeat-SlashEvent” in Internals Reference, Section “DoubleRepeatSlash” in Internals Reference, Section “PercentRepeat” in Internals Reference, Section “PercentRepeatCounter” in Internals Reference, Section “PercentRepeatedMusic” in Internals Reference, Section “Percent_repeat_engraver” in Internals Reference, Section “DoublePercentEvent” in Internals Reference, Section “DoublePercentRepeat” in Internals Reference, Section “DoublePercentRepeatCounter” in Internals Reference, Section “Double_percent_repeat_engraver” in Internals Reference.

Known issues and warnings

Percent repeats will not contain anything else apart from the percent sign itself; in particular, timing changes will not be repeated.

\begin{verbatim}
\repeat percent 3 { \time 5/4 c2. 2 \time 4/4 2 2 }
\end{verbatim}

Any meter changes or \texttt{\textbackslash partial} commands need to occur in parallel passages \textit{outside} of any percent repeat, e.g in a separate timing track.

\begin{verbatim}
<< \repeat percent 3 { c2. 2 2 2 } \repeat unfold 3 { \time 5/4 s4*5 \time 4/4 s1 } >>
\end{verbatim}
Tremolo repeats

Tremolos can take two forms: alternation between two chords or two notes, and rapid repetition of a single note or chord. Tremolos consisting of an alternation are indicated by adding beams between the notes or chords being alternated, while tremolos consisting of the rapid repetition of a single note are indicated by adding beams or slashes to a single note.

To place tremolo marks between notes, use \repeat with tremolo style:

\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}

\repeat tremolo
\timesig.C44\clefs.G\noteheads.s2\dots.dot\noteheads.s1\noteheads.s2/\noteheads.s0/timesig.C44/clefs.G/noteheads.s0

The \repeat tremolo syntax expects exactly two notes within the braces, and the number of repetitions must correspond to a note value that can be expressed with plain or dotted notes. Thus, \repeat tremolo 7 is valid and produces a double dotted note, but \repeat tremolo 9 is not.

The duration of the tremolo equals the duration of the braced expression multiplied by the number of repeats: \repeat tremolo 8 { c16 d16 } gives a whole note tremolo, notated as two whole notes joined by tremolo beams.

There are two ways to put tremolo marks on a single note. The \repeat tremolo syntax is also used here, in which case the note should not be surrounded by braces:

\repeat tremolo 4 c'16

\repeat tremolo
\timesig.C44\clefs.G\noteheads.s1\noteheads.s0/timesig.C44/clefs.G/noteheads.s0

The same output can be obtained by adding :N after the note, where N indicates the duration of the subdivision (it must be at least 8). If N is 8, one beam is added to the note’s stem. If N is omitted, the last value is used:

\relative {
  c''2:8 c:32
  c: c:
}

\select/N
c'\prime\prime2:8 c:\prime32
c: c:

Selected Snippets

Cross-staff tremolos

Since \repeat tremolo expects exactly two musical arguments for chord tremolos, the note or chord which changes staff within a cross-staff tremolo should be placed inside curly braces together with its \change Staff command.

\new PianoStaff <<
\new Staff = "up" \relative c'' { 
  \key a \major 
  \time 3/8 
  s4. 
} 
\new Staff = "down" \relative c'' { 
  \key a \major 
  \time 3/8 
  \voiceOne 
  \repeat tremolo 6 { 
    <a e'>32 
      \change Staff = "up" 
      \voiceTwo 
      <cis a' dis>32 
  } 
} 
} 
>>

See also

Snippets: Section “Repeats” in Snippets.

1.5 Simultaneous notes
Polyphony in music refers to having more than one voice occurring in a piece of music. Polyphony in LilyPond refers to having more than one voice on the same staff.

1.5.1 Single voice
This section discusses simultaneous notes inside the same voice.

Chorded notes
A chord is formed by enclosing a set of pitches between < and >. A chord may be followed by a duration just like simple notes.

```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```

Chords may also be followed by articulations, again just like simple notes.

```
\relative {
  <a' c e>1 \fermata <a c e>2-> <f a c e>4 \prall <a c>8. ^! <g c e>16-. 
}
```

The notes within the chord themselves can also be followed by articulation and ornamentation.

```
\relative {
  <a' c> \prall e>1 <a-> c>2 <f-. a c-. e-.>4
}
```
However some notation, such as dynamics and hairpins must be attached to the chord rather than to notes within the chord, otherwise they will not print. Other notation like fingerings and slurs will get placed markedly different when attached to notes within a chord rather than to whole chords or single notes.

\relative {
\grace { g'8( a b }
\markup \italic "sempre staccato"
\repeat unfold 4 { c4 e } c1\f
}

Relative mode can be used for pitches in chords. The first note of each chord is always relative to the first note of the chord that came before it, or in the case where no preceding chord exists, the pitch of the last note that came before the chord. All remaining notes in the chord are relative to the note that came before it within the same chord.

\relative {
\grace { a' c e}1 <f a c> <a c e> <f' a c> <b, e b,>

For more information about chords, see Section 2.7 [Chord notation], page 423.
See also

Notation Reference: Section 2.7 [Chord notation], page 423, [Articulations and ornamentations], page 122, [Relative octave entry], page 2, Section 1.5.2 [Multiple voices], page 173.
Snippets: Section “Simultaneous notes” in Snippets.

Known issues and warnings

Chords containing more than two pitches within a staff space, such as ‘<e f! fis!>’, create overlapping noteheads. Depending on the situation, better representations might involve
- temporary use of Section 1.5.2 [Multiple voices], page 173, ‘<< f! \ <e fis! >>’,
- enharmonic transcription of one or more pitches, ‘<e f ges>’, or
- [Clusters], page 173.

Chord repetition

In order to save typing, a shortcut can be used to repeat the preceding chord. The chord repetition symbol is q:

\relative {<a’ c e>1 q <f a c>2 q}

\relative {<a’ c e>1 \p^"text" q2<( q8)[-! q8.]\! q16-1-2-3 q8\prall}

As with regular chords, the chord repetition symbol can be used with durations, articulations, markups, slurs, beams, etc., as only the pitches of the previous chord are duplicated.

\relative {<a’ c e>1 c’4 q2 r8 q8 | q2 c, |}

The chord repetition symbol always remembers the last instance of a chord so it is possible to repeat the most recent chord even if other non-chorded notes or rests have been added since.
However, the chord repetition symbol does not retain any dynamics, articulation or ornamentation within, or attached to, the previous chord.

```
relative {
  <a'-. c\prall e>1\sfz c'4 q2 r8 q8 | q2 c, |}
```

To have some of them retained, the `\chordRepeats` function can be called explicitly with an extra argument specifying a list of event types to keep unless events of that type are already present on the q chord itself.

```
relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } | q2 c, |}
```

Here using `\chordRepeats` inside of a `\relative` construction produces unexpected results: once chord events have been expanded, they are indistinguishable from having been entered as regular chords, making `\relative` assign an octave based on their current context.

Since nested instances of `\relative` don’t affect one another, another `\relative` inside of `\chordRepeats` can be used for establishing the octave relations before expanding the repeat chords. In that case, the whole content of the inner `\relative` does not affect the outer one; hence the different octave entry of the final note in this example.

```
relative {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } | q2 c'" | }
```

Interactions with `\relative` occur only with explicit calls of `\chordRepeats`: the implicit expansion at the start of typesetting is done at a time where all instances of `\relative` have already been processed.
Simultaneous expressions

One or more music expressions enclosed in double angle brackets are taken to be simultaneous. If the first expression begins with a single note or if the whole simultaneous expression appears explicitly within a single voice, the whole expression is placed on a single staff; otherwise the elements of the simultaneous expression are placed on separate staves.

The following examples show simultaneous expressions on one staff:

\begin{verbatim}
\new Voice { \% explicit single voice
  << \relative { a'4 b g2 } \relative { d'4 g c,2 } >>
}
\end{verbatim}

This can be useful if the simultaneous sections have identical rhythms, but attempts to attach notes with different durations to the same stem will cause errors. Notes, articulations, and property changes in a single ‘Voice’ are collected and engraved in musical order:

\begin{verbatim}
\relative {
  <a' c>4-. <>-. << c a >> << { c-. <c a> } { a s-. } >>
}
\end{verbatim}

Multiple stems or beams or different note durations or properties at the same musical time require the use of multiple voices.

The following example shows how simultaneous expressions can generate multiple staves implicitly:

\begin{verbatim}
\% no single first note
<< \relative { a'4 b g2 } \relative { d'4 g2 c,4 } >>
\end{verbatim}
Here different rhythms cause no problems because they are interpreted in different voices.

**Known issues and warnings**

If notes from two or more voices, with no shifts specified, have stems in the same direction, the message

```
warning: This voice needs a \voiceXx or \shiftXx setting
```

will appear during compilation. This message can be suppressed by:

```
\override NoteColumn.ignore-collision = ##t
```

However, this not only suppresses the warning but will prevent any collision resolution whatsoever and may have other unintended effects (also see **Known Issues** in [Collision resolution], page 178).

**Clusters**

A cluster indicates a continuous range of pitches to be played. They can be denoted as the envelope of a set of notes. They are entered by applying the function `\makeClusters` to a sequence of chords, e.g.,

```
\relative \makeClusters { <g' b>2 <c g'> }
```

Ordinary notes and clusters can be put together in the same staff, even simultaneously. In such a case no attempt is made to automatically avoid collisions between ordinary notes and clusters.

**See also**

- Snippets: Section “Simultaneous notes” in *Snippets*.

**Known issues and warnings**

Clusters look good only if they span at least two chords; otherwise they appear too narrow.

Clusters do not have a stem and cannot indicate durations by themselves, but the length of the printed cluster is determined by the durations of the defining chords. Separate clusters need a separating rest between them.

Clusters do not produce MIDI output.

**1.5.2 Multiple voices**

This section discusses simultaneous notes in multiple voices or multiple staves.

**Single-staff polyphony**
Explicitly instantiating voices

The basic structure needed to achieve multiple independent voices in a single staff is illustrated in the following example:

\new Staff <<
\new Voice = "first"
\relative { \voiceOne r8 r16 g'' e8. f16 g8[ c,] f e16 d }
\new Voice = "second"
\relative { \voiceTwo d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>

Here, voices are instantiated explicitly and are given names. The \voiceOne ... \voiceFour commands set up the voices so that first and third voices get stems up, second and fourth voices get stems down, third and fourth voice note heads are horizontally shifted, and rests in the respective voices are automatically moved to avoid collisions. The \oneVoice command returns all the voice settings to the neutral default directions.

Temporary polyphonic passages

A temporary polyphonic passage can be created with the following construct:

<< { \voiceOne ... }
\new Voice { \voiceTwo ... }
>> \oneVoice

Here, the first expression within a temporary polyphonic passage is placed into the Voice context which was in use immediately before the polyphonic passage, and that same Voice context continues after the temporary section. Other expressions within the angle brackets are assigned to distinct temporary voices. This allows lyrics to be assigned to one continuing voice before, during and after a polyphonic section:

\relative <<
\new Voice = "melody" {
a'4
<<
{ \voiceOne
  g f
}
\new Voice {
  \voiceTwo
  d2
}
>>
\oneVoice
e4
}
\new Lyrics \lyricsto "melody" {
This is my song.
}
>>
Here, the \voiceOne and \voiceTwo commands are required to define the settings of each voice.

**The double backslash construct**

The \<< {...} \ \ {...} >> construct, where the two (or more) expressions are separated by double backslashes, behaves differently to the similar construct without the double backslashes: *all* the expressions within this construct are assigned to new Voice contexts. These new Voice contexts are created implicitly and are given the fixed names "1", "2", etc.

The first example could be typeset as follows:

\<<
\relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
\\%
\relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
\>>

This syntax can be used where it does not matter that temporary voices are created and then discarded. These implicitly created voices are given the settings equivalent to the effect of the \voiceOne . . . \voiceFour commands, in the order in which they appear in the code.

In the following example, the intermediate voice has stems up, therefore we enter it in the third place, so it becomes voice three, which has the stems up as desired. Spacer rests are used to avoid printing doubled rests.

\<<
\relative { r8 g'' g g g f16 ees f8 d }
\\%
\relative { ees'8 r ees r d r d r }
\\%
\relative { d''8 s c s bes s a s }
\>>

In all but the simplest works it is advisable to create explicit Voice contexts as explained in Section “Contexts and engravers” in Learning Manual and Section “Explicitly instantiating voices” in Learning Manual.

**Voice order**

When entering multiple voices in the input file, use the following order:

- **Voice 1**: highest
- **Voice 2**: lowest
- **Voice 3**: second highest
Voice 4: second lowest
Voice 5: third highest
Voice 6: third lowest
etc.

Though this may seem counterintuitive, it simplifies the automatic layout process. Note that the odd-numbered voices are given upstems, and the even-numbered voices are given downstems:

```
\new Staff <<
\time 2/4
{ f''2 } % 1: highest
\{}
{ c'2 } % 2: lowest
\{}
{ d''2 } % 3: second-highest
\{}
{ e'2 } % 4: second-lowest
\{}
{ b'2 } % 5: third-highest
\{}
{ g'2 } % 6: third-lowest
>>
```

When a different voice entry order is desired, the command `\voices` may be convenient:

```
\new Staff \voices 1,3,5,6,4,2 <<
\time 2/4
{ f''2 } % 1: highest
\{}
{ d''2 } % 3: second-highest
\{}
{ b'2 } % 5: third-highest
\{}
{ g'2 } % 6: third-lowest
\{}
{ e'2 } % 4: second-lowest
\{}
{ c'2 } % 2: lowest
>>
```

**Note:** Lyrics and spanners (such as slurs, ties, hairpins, etc.) cannot be created 'across' voices.
Identical rhythms

In the special case that we want to typeset parallel pieces of music that have the same rhythm, we can combine them into a single Voice context, thus forming chords. To achieve this, enclose them in a simple simultaneous music construct within an explicit voice:

\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>

This method leads to strange beamings and warnings if the pieces of music do not have the same rhythm.

Predefined commands

\voiceOne, \voiceTwo, \voiceThree, \voiceFour, \oneVoice.

See also


Notation Reference: [Percussion staves], page 404, [Invisible rests], page 61, [Stems], page 234.

Snippets: Section “Simultaneous notes” in Snippets.

Voice styles

Voices may be given distinct colors and shapes, allowing them to be easily identified:

<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \relative { \voiceTwoStyle e'2 e }
  \relative { \voiceThreeStyle b2. c4 }
  \relative { \voiceFourStyle g'2 g }
>>

The \voiceNeutralStyle command is used to revert to the standard presentation.

Predefined commands

\voiceOneStyle, \voiceTwoStyle, \voiceThreeStyle, \voiceFourStyle, \voiceNeutralStyle.

See also


Snippets: Section “Simultaneous notes” in Snippets.
Collision resolution

The note heads of notes in different voices with the same pitch, same note head and opposite stem direction are automatically merged, but notes with different note heads or the same stem direction are not. Rests opposite a stem in a different voice are shifted vertically. The following example shows three different circumstances, on beats 1 and 3 in bar 1 and beat 1 in bar 2, where the automatic merging fails.

```
<<
  \relative {  
    c''8 d e d c d c4  
    g'2 fis  
  } \ \  
  \relative {  
    c''2 c8. b16 c4  
    e,2 r  
  } \ \  
  \relative {  
    \oneVoice  
    s1  
    e'8 a b c d2  
  } 
>>
```

Notes with different note heads may be merged as shown below. In this example the note heads on beat 1 of bar 1 are now merged:

```
<<
  \relative {  
    \mergeDifferentlyHeadedOn  
    c''8 d e d c d c4  
    g'2 fis  
  } \ \  
  \relative {  
    c''2 c8. b16 c4  
    e,2 r  
  } \ \  
  \relative {  
    \oneVoice  
    s1  
    e'8 a b c d2  
  } 
>>
```

Quarter and half notes are not merged in this way, since it would be difficult to tell them apart.
Note heads with different dots as shown in beat 3 of bar 1 may be also be merged:

```
<<
relative {
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  c''8 d e d c d c4
g'2 fis
}
\ relative {
  c''2 c8. b16 c4
e,2 r
} \
\relative {
  \oneVoice
  s1
e'8 a b c d2
}
>>
```

The half note and eighth note at the start of the second measure are incorrectly merged because the automatic merge cannot successfully complete the merge when three or more notes line up in the same note column, and in this case the merged note head is incorrect. To allow the merge to select the correct note head a `\shift` must be applied to the note that should not be merged. Here, `\shiftOn` is applied to move the top g out of the column, and `\mergeDifferentlyHeadedOn` then works properly.

```
<<
relative {
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  c''8 d e d c d c4
  \shiftOn
g'2 fis
}
\ relative {
  c''2 c8. b16 c4
e,2 r
} \
\relative {
  \oneVoice
  s1
e'8 a b c d2
}
>>
```
The \shiftOn command allows (but does not force) the notes in a voice to be shifted. When \shiftOn is applied to a voice, a note or chord in that voice is shifted only if its stem would otherwise collide with a stem from another voice, and only if the colliding stems point in the same direction. The \shiftOff command prevents this type of shifting from occurring.

By default, the outer voices (normally voices one and two) have \shiftOff specified, while the inner voices (three and above) have \shiftOn specified. When a shift is applied, voices with upstems (odd-numbered voices) are shifted to the right, and voices with downstems (even-numbered voices) are shifted to the left.

Here is an example to help you visualize how an abbreviated polyphonic expression would be expanded internally.

Note: Note that with three or more voices, the vertical order of voices in your input file should not be the same as the vertical order of voices on the staff!

\new Staff \relative { 
%%% abbreviated entry 
<<
  \{ f''2 \} % 1: highest
  \{ g,2 \} % 2: lowest
  \{ d'2 \} % 3: upper middle
  \{ b2 \} % 4: lower middle
>>
%%% internal expansion of the above 
<<
  \new Voice = "1" { \voiceOne \shiftOff f'2 }
  \new Voice = "2" { \voiceTwo \shiftOff g,2 }
  \new Voice = "3" { \voiceThree \shiftOn d'2 } % shifts right
  \new Voice = "4" { \voiceFour \shiftOn b2 } % shifts left
>> 
}

Two additional commands, \shiftOnn and \shiftOnnn provide further shift levels which may be specified temporarily to resolve collisions in complex situations – see Section “Real music example” in Learning Manual.

Notes are only merged if they have opposing stem directions (as they have, for example, in voices one and two by default or when the stems are explicitly set in opposite directions).

Predefined commands
\mergeDifferentlyDottedOn, \mergeDifferentlyDottedOff, \mergeDifferentlyHeadedOn, \mergeDifferentlyHeadedOff.
\shiftOn, \shiftOnn, \shiftOnnn, \shiftOff.
Selected Snippets

Additional voices to avoid collisions

In some instances of complex polyphonic music, additional voices are necessary to prevent collisions between notes. If more than four parallel voices are needed, additional voices can be added by defining a variable using the Scheme function `context-spec-music`.

```scheme
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)
```

```
\relative c' {
  \time 3/4
  \key d \minor
  \partial 2
  \new Voice {
    \voiceOne
    a4. a8
    e'4 e4. e8
    f4 d4. c8
  }
  \new Voice {
    \voiceTwo
    d,2
d4 cis2
d4 bes2
  }
  \new Voice {
    \voiceThree
    f'2
    bes4 a2
    a4 s2
  }
  \new Voice {
    \voiceFive
    s2
g4 g2
    f4 f2
  }

  \new Staff \relative c' <<
  f2. f4
}
```

Moving dotted notes in polyphony

When a dotted note in the upper voice is moved to avoid a collision with a note in another voice, the default is to move the upper note to the right. This behaviour can be over-ridden by using the `prefer-dotted-right` property of `NoteCollision`.
Forcing horizontal shift of notes

When the typesetting engine cannot cope, the following syntax can be used to override typesetting decisions. The units of measure used here are staff spaces.

\relative c' <<
{ 
  \( \langle d\ g\rangle \) ^2 \langle d\ g\rangle
}
\( \langle b\ f'\rangle \) ^2
\once \override NoteColumn.force-hshift = #1.7
\langle b\ f'\rangle ^2
>>

See also

Snippets: Section “Simultaneous notes” in Snippets.
Internals Reference: Section “NoteColumn” in Internals Reference, Section “NoteCollision” in Internals Reference, Section “RestCollision” in Internals Reference.

Known issues and warnings

Using \override NoteColumn.ignore-collision = ##t will cause differently headed notes in different voices to merge incorrectly.
Merging rests

When using multiple voices it is common to merge rests which occur in both parts. This can be accomplished using `Merge_rests_engraver`.

```plaintext
voiceA = \relative { d''4 r d2 | R1 | }
voiceB = \relative { fis'4 r g2 | R1 | }
\score {
  <<
    \new Staff \with {
      instrumentName = "unmerged"
    }
    \new Voice { \voiceOne \voiceA }
    \new Voice { \voiceTwo \voiceB }
  >>
  \new Staff \with {
    instrumentName = "merged"
    \consists "Merge_rests_engraver"
  }
  \new Voice { \voiceOne \voiceA }
  \new Voice { \voiceTwo \voiceB }
} >>
```

Setting the context property `suspendRestMerging` to `##t` allows for turning off rest merging temporarily.

Automatic part combining

Automatic part combining is used to merge two separate parts of music onto a single staff. This can be especially helpful when typesetting orchestral scores. A single `Voice` is printed while the two parts of music are the same, but in places where they differ, a second `Voice` is printed. Stem directions are set up & down accordingly while Solo and a due parts are also identified and marked appropriately.

The syntax for automatic part combining is:

```plaintext
\partCombine musicexpr1 musicexpr2
```

The following example demonstrates the basic functionality, putting parts on a single staff as polyphony and setting stem directions accordingly. The same variables are used for the independent parts and the combined staff.

```plaintext
instrumentOne = \relative {
  c'4 d e f |
```
Both parts have identical notes in the third measure, so only one instance of the notes is printed. Stem, slur, and tie directions are set automatically, depending on whether the parts are playing solo or in unison. When needed in polyphony situations, the first part (with context called one) gets “up” stems, while the second (called two) always gets “down” stems. In solo situations, the first and second parts get marked with “Solo” and “Solo II”, respectively. The unison (a due) parts are marked with the text “a2”.

By default, the partCombiner merges two notes of the same pitch as an a due note, combines notes with the same rhythm less than a ninth apart as chords and separates notes more than a ninth apart (or when the voices cross) into separate voices. This can be overridden with an optional argument of a pair of numbers after the \partCombine command: the first specifies the interval where notes start to be combined (the default is zero) and the second where the notes are split into separate voices. Setting the second argument to zero means that the partCombiner splits notes with an interval of a second or more, setting it to one splits notes of a third or more, and so on.

\instrumentOne = \relative {
    a4 b c d |
    e f g a |
    b c d e |
}
Both arguments to `\partCombine` will be interpreted as separate Voice contexts, so if the music is being specified in relative mode then both parts must contain a `\relative` function, i.e.,

```
\partCombine
  \relative ... musicexpr1
  \relative ... musicexpr2
```

A `\relative` section that encloses a `\partCombine` has no effect on the pitches of `musicexpr1` or `musicexpr2`.

In professional scores, voices are often kept apart from each other for long passages of music even if some of the notes are the same in both voices, and could just as easily be printed as unison. Combining notes into a chord, or showing one voice as solo is, therefore, not ideal as the `\partCombine` function considers each note separately. In this case the `\partCombine` function can be overridden with one of the following commands. All of the commands may be preceded with `\once` in order to have them only apply to the next note in the music expression.

- `\partCombineApart` keeps the notes as two separate voices, even if they can be combined into a chord or unison.
- `\partCombineChords` combines the notes into a chord.
- `\partCombineUnisono` combines both voices as “unison”.
- `\partCombineSoloI` prints only voice one, and marks it as a “Solo”.
- `\partCombineSoloII` prints only voice two and marks it as a “Solo”.
- `\partCombineAutomatic` ends the functions of the commands above, and reverts back to the standard `\partCombine` functionality.

```
instrumentOne = \relative c' { 
  \partCombineApart c2"apart" e | 
  \partCombineAutomatic e2"auto" e | 
  \partCombineChords e2"chord" e | 
  \partCombineAutomatic c2"auto" c | 
  \partCombineApart c2"apart" \once \partCombineChords e"chord once" | 
  c2 c |
} 

instrumentTwo = \relative {
```
Using `\partCombine` with lyrics

The `\partCombine` command is not designed to work with lyrics; if one of the voices is explicitly named in order to attach lyrics to it, the partCombiner will stop working. However, this effect can be achieved using a `NullVoice` context. See [Polyphony with shared lyrics], page 304.

Selected Snippets

Combining two parts on the same staff

The part combiner tool ( `\partCombine` command ) allows the combination of several different parts on the same staff. Text directions such as “solo” or “a2” are added by default; to remove them, simply set the property `printPartCombineTexts` to `f`. For vocal scores (hymns), there is no need to add “solo/a2” texts, so they should be switched off. However, it might be better not to use it if there are any solos, as they won’t be indicated. In such cases, standard polyphonic notation may be preferable.

This snippet presents the three ways two parts can be printed on a same staff: standard polyphony, `\partCombine` without texts, and `\partCombine` with texts.

```
%% Combining pedal notes with clef changes

musicUp = \relative c'' { 
    \time 4/4 
    a4 c4.( g8) a4 | 
    g4 e' g,( a8 b) | 
    c b a2. 
} 
```
-changing partcombine texts

When using the automatic part combining feature, the printed text for the solo and unison sections may be changed:

\new Staff <<
\set Staff.soloText = "girl"
\set Staff.soloIIText = "boy"
\set Staff.aDueText = "together"

\score {<<
\new Staff \with { instrumentName = "Standard polyphony" }
<< \musicUp \\ \musicDown >>
\new Staff \with {
instrumentName = "PartCombine without text"
printPartCombineTexts = ##f
}
\partCombine \musicUp \musicDown
\new Staff \with { instrumentName = "PartCombine with text" }
\partCombine \musicUp \musicDown
>>
\layout {
indent = 6.0\cm
\context {
\Score
\override SystemStartBar.collapse-height = #30
}
}
\partCombine
\relative c' {
  g4 g r r
  a2 g
}
\relative c' {
  r4 r a( b)
  a2 g
}

\begin{music}
\begin{staff}
\begin{noteheads.s1}
\begin{noteheads.s2}
\begin{noteheads.s2}
\timesig.C44/clefs.G
\end{noteheads.s2}
\end{noteheads.s2}
\end{noteheads.s1}
\end{music}
\end{music}

\begin{music}
\begin{staff}
\begin{noteheads.s1}
\begin{noteheads.s2}
\begin{noteheads.s2}
\timesig.C44/clefs.G
\end{noteheads.s2}
\end{noteheads.s2}
\end{noteheads.s1}
\end{music}

See also

Notation Reference: Section 1.6.3 [Writing parts], page 210.
Snippets: Section “Simultaneous notes” in Snippets.

Known issues and warnings

All \partCombine... functions can only accept two voices.
\partCombine... functions cannot be placed inside a \tuplet or \relative block.

If printPartCombineTexts is set and the two voices play the same notes “on and off”, in the same measure, the part combiner may typeset a2 more than once in that measure.

\partCombine only knows when a note starts in a Voice; it cannot, for example, remember if a note in one Voice has already started when combining notes that have just started in the other Voice. This can lead to a number of unexpected issues including “Solo” or “Unison” marks being printed incorrectly.

\partCombine keeps all spanners (slurs, ties, hairpins, etc.) in the same Voice so that if any such spanners start or end in a different Voice, they may not be printed properly or at all.

If the \partCombine function cannot combine both music expressions (i.e., when both voices have different durations), it will give the voices, internally, its own custom names: one and two respectively. This means if there is any “switch” to a differently named Voice context, the events in that differently named Voice will be ignored.

Refer also to Known issues and warnings when using \partCombine with tablature in [Default tablatures], page 355, and the Note in [Automatic beams], page 84, when using automatic beaming.

Writing music in parallel

Music for multiple parts can be interleaved in input code. The function \parallelMusic accepts a list with the names of a number of variables to be created, and a musical expression. The content of alternate measures from the expression become the value of the respective variables, so you can use them afterwards to print the music.
Note: Bar checks | must be used, and the measures must be of the same length.

`\parallelMusic voiceA,voiceB,voiceC {`
```
% Bar 1
r8 g'16 c'' e'' g' c'' e'' g' c'' e'' |
r16 e'8.~ 4 r16 e'8.~ 4 |
c'2 c'2 |
```
% Bar 2
```
r8 a'16 d'' f'' a' d'' f'' a' d'' f'' |
r16 d'8.~ 4 r16 d'8.~ 4 |
c'2 c'2 |
```
}
\new StaffGroup <<
\new Staff << \relative c'' \voiceA \relative c' \voiceB >>
\new Staff { \clef bass \voiceC }
>>

Relative mode may be used. Note that the `\relative` command is not used inside `\parallelMusic` itself. The notes are relative to the preceding note in the voice, not to the previous note in the input – in other words, relative notes for `voiceA` ignore the notes in `voiceB`.

`\parallelMusic voiceA,voiceB,voiceC {`
```
% Bar 1
r8 g16 c e g, c e r8 g,16 c e g, c e |
r16 e8.~ 4 r16 e8.~ 4 |
c2 c |
```
% Bar 2
```
r8 a,16 d f a, d f r8 a,16 d f a, d f |
r16 d8.~ 4 r16 d8.~ 4 |
c2 c |
```
}
\new StaffGroup <<
\new Staff << \relative c'' \voiceA \relative c' \voiceB >>
\new Staff \relative c' { \clef bass \voiceC }
>>
This works quite well for piano music. This example maps four consecutive measures to four variables:

```latex
\begin{verbatim}
\global = {
    \key g \major
    \time 2/4
}

\parallelMusic voiceA,voiceB,voiceC,voiceD {
    % Bar 1
    a8 b c d |
    d4 e   |
    c16 d e fis d e fis g |
    a4 a   |

    % Bar 2
    e8 fis g a |
    fis4 g |
    e16 fis g a fis g a b |
    a4 a   |

    % Bar 3 ...
}

\score {
    \new PianoStaff <<
        \new Staff {
            \global <<
                \relative c'' \voiceA
                \relative c' \voiceB
            >>
        }
        \new Staff {
            \global \clef bass <<
                \relative c \voiceC
                \relative c \voiceD
            >>
        }
    >>
}
\end{verbatim}
```

![Musical notation diagram]
See also
Snippets: Section “Simultaneous notes” in Snippets.

1.6 Staff notation

This section explains how to influence the appearance of staves, how to print scores with more than one staff, and how to add tempo indications and cue notes to staves.

1.6.1 Displaying staves
This section describes the different methods of creating and grouping staves.

Instantiating new staves
Staves (singular: staff) are created with the \new or \context commands. For details, see Section 5.1.2 [Creating and referencing contexts], page 599.

The basic staff context is Staff:
\new Staff \relative { c''4 d e f }

The DrumStaff context creates a five-line staff set up for a typical drum set. Each instrument is shown with a different symbol. The instruments are entered in drum mode following a \drummode command, with each instrument specified by name. For details, see [Percussion staves], page 404.
\new DrumStaff {
Chapter 1: Musical notation

\drummode { cymc hh ss tohm }
}

RhythmicStaff creates a single-line staff that only displays the rhythmic values of the input. Real durations are preserved. For details, see [Showing melody rhythms], page 82.
\new RhythmicStaff { c4 d e f }

TabStaff creates a tablature with six strings in standard guitar tuning. For details, see [Default tablatures], page 355.
\new TabStaff \relative { c''4 d e f }

There are two staff contexts specific for the notation of ancient music: MensuralStaff and VaticanaStaff. They are described in [Pre-defined contexts], page 448.

The GregorianTranscriptionStaff context creates a staff to notate modern Gregorian chant. It does not show bar lines.
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }

New single staff contexts may be defined. For details, see Section 5.1.6 [Defining new contexts], page 612.

See also


Notation Reference: Section 5.1.2 [Creating and referencing contexts], page 599, [Percussion staves], page 404, [Showing melody rhythms], page 82, [Default tablatures], page 355, [Pre-defined contexts], page 448, [Staff symbol], page 199, [Gregorian chant contexts], page 457, [Mensural contexts], page 450, Section 5.1.6 [Defining new contexts], page 612.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “Staff” in Internals Reference, Section “DrumStaff” in Internals Reference, Section “GregorianTranscriptionStaff” in Internals Reference, Section “RhythmicStaff” in Internals Reference, Section “TabStaff” in Internals Reference, Section “MensuralStaff” in Internals Reference, Section “VaticanaStaff” in Internals Reference, Section “StaffSymbol” in Internals Reference.
Grouping staves

Various contexts exist to group single staves together in order to form multi-staff systems. Each grouping context sets the style of the system start delimiter and the behavior of bar lines.

If no context is specified, the default properties will be used: the group is started with a vertical line, and the bar lines are not connected.

```
<<
\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
>>
```

In the **StaffGroup** context, the group is started with a bracket and bar lines are drawn through all the staves.

```
\new StaffGroup <<
\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
>>
```

In a **ChoirStaff**, the group starts with a bracket, but bar lines are not connected.

```
\new ChoirStaff <<
\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
>>
```

In a **GrandStaff**, the group begins with a brace, and bar lines are connected between the staves.

```
\new GrandStaff <<
\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
>>
```
The **PianoStaff** is identical to a **GrandStaff**, except that it supports printing the instrument name directly. For details, see [Instrument names], page 210.

```latex
\new PianoStaff \with { instrumentName = "Piano" }
<<
\new Staff \relative { c''1 c }
\new Staff \relative { \clef bass c1 c }
>>
```

Each staff group context sets the property `systemStartDelimiter` to one of the following values: `SystemStartBar`, `SystemStartBrace`, or `SystemStartBracket`. A fourth delimiter, `SystemStartSquare`, is also available, but it must be explicitly specified.

New staff group contexts may be defined. For details, see Section 5.1.6 [Defining new contexts], page 612.

### Selected Snippets

**Use square bracket at the start of a staff group**

The system start delimiter `SystemStartSquare` can be used by setting it explicitly in a `StaffGroup` or `ChoirStaff` context.

```latex
\score { 
\new StaffGroup { <<
\set StaffGroup.systemStartDelimiter = #'SystemStartSquare
\new Staff { c'4 d' e' f' }
\new Staff { c'4 d' e' f' }
>> }
}
```

**Display bracket with only one staff in a system**

If there is only one staff in one of the staff types `ChoirStaff` or `StaffGroup`, by default the bracket and the starting bar line will not be displayed. This can be changed by overriding `collapse-height` to set its value to be less than the number of staff lines in the staff.
Note that in contexts such as PianoStaff and GrandStaff where the systems begin with a brace instead of a bracket, another property has to be set, as shown on the second system in the example.

\score {\
  \new StaffGroup << \\%
  % Must be lower than the actual number of staff lines
  \override StaffGroup.SystemStartBracket.collapse-height = #4 \\
  \override Score.SystemStartBar.collapse-height = #4 \\
  \new Staff { c'1 \\
  } \\
  >> \\
} \\
\score {\
  \new PianoStaff << \\
  \override PianoStaff.SystemStartBrace.collapse-height = #4 \\
  \override Score.SystemStartBar.collapse-height = #4 \\
  \new Staff { c'1 \\
  } \\
  >> \\
} \\

\begin{music}
\relative c'' { \\
  \global { \hide Staff.BarLine s1 s \\
    % the final bar line is not interrupted \\
    \undo \hide Staff.BarLine \\
    \bar { "\} \\
  } \\
  \new StaffGroup \relative c'' { \\
    << \\
    \new Staff { << \global { c1 c } >> } \\
    \new Staff { << \global { c c } >> } \\
    >> \\
  } \\
}
\end{music}

*Mensurstriche layout (bar lines between the staves)*

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a StaffGroup instead of a ChoirStaff. The bar line on staves is blanked out using \hide.

\global { \\
  \hide Staff.BarLine s1 s \\
  % the final bar line is not interrupted \\
  \undo \hide Staff.BarLine \\
  \bar { "\} \\
} \\

\new StaffGroup \relative c'' { \\
  << \\
  \new Staff { << \global { c1 c } >> } \\
  \new Staff { << \global { c c } >> } \\
  >> \\
}
Chapter 1: Musical notation

See also


Notation Reference: [Instrument names], page 210, Section 5.1.6 [Defining new contexts], page 612.

Snippets: Section “Staff notation” in *Snippets*.


Nested staff groups

Staff-group contexts can be nested to arbitrary depths. In this case, each child context creates a new bracket adjacent to the bracket of its parent group.

```latex
\new StaffGroup <<
\new Staff \relative { c'2 c | c2 c }
\new StaffGroup <<
\new Staff \relative { g'2 g | g2 g }
\new StaffGroup \with {
  systemStartDelimiter = #'SystemStartSquare
}
<<
\new Staff \relative { e'2 e | e2 e }
\new Staff \relative { c'2 c | c2 c }
>>
>>
```

New nested staff group contexts can be defined. For details, see Section 5.1.6 [Defining new contexts], page 612.
Nesting staves

The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The command `<\set StaffGroup.systemStartDelimiterHierarchy>` takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```latex
\new StaffGroup
\relative c' <<
  \override StaffGroup.SystemStartSquare.collapse-height = \#4
  \set StaffGroup.systemStartDelimiterHierarchy
  = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                                   (SystemStartSquare b) ) c ) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```

See also

Notation Reference: [Grouping staves], page 193, [Instrument names], page 210, Section 5.1.6 [Defining new contexts], page 612.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “StaffGroup” in Internals Reference, Section “ChoirStaff” in Internals Reference, Section “SystemStartBar” in Internals Reference, Section “SystemStartBrace” in Internals Reference, Section “SystemStartBracket” in Internals Reference, Section “SystemStartSquare” in Internals Reference.
Separating systems

If the number of systems per page changes from page to page it is customary to separate the systems by placing a system separator mark between them. By default the system separator is blank, but can be turned on with a \paper option.

\book{
  \score{
    \new StaffGroup <<
    \new Staff {
      \relative {
        c''4 c c c 
        \break
        c4 c c c
      }
    }
    \new Staff {
      \relative {
        c''4 c c c 
        \break
        c4 c c c
      }
    }
  }>>
  \paper{system-separator-markup = \slashSeparator
      % following commands are needed only to format this documentation
      paper-width = 100\mm
      paper-height = 100\mm
      tagline = ##f
  }
}

See also

Notation Reference: Section 4.1 [Page layout], page 546.
Snippets: Section “Staff notation” in Snippets.
1.6.2 Modifying single staves

This section explains how to change specific attributes of one staff: for example, modifying the number of staff lines or the staff size. Methods to start and stop staves and set ossia sections are also described.

Staff symbol

The \stopStaff and \startStaff commands can be used to stop or (re)start the staff lines respectively, from being printed at any point within a score.

\relative {
    \stopStaff f''4 d \startStaff g, e
    f'4 d \stopStaff g, e
    f'4 d \startStaff g, e
}\n
Predefined commands

\startStaff, \stopStaff. The lines of a staff belong to the StaffSymbol grob (including ledger lines) and can be modified using StaffSymbol properties, but these modifications must be made before the staff is (re)started.

The number of staff lines can be altered:

\relative {
    f''4 d \stopStaff
    \override Staff.StaffSymbol.line-count = #2
    \startStaff g, e |
    f'4 d \stopStaff
    \revert Staff.StaffSymbol.line-count
    \startStaff g, e |
}\n
The position of each staff line can also be altered. A list of numbers sets each line's position. 0 corresponds to the normal center line, and the normal line positions are (-4 -2 0 2 4). A single staff line is printed for every value entered so that the number of staff lines, as well as their position, can be changed with a single override.

\relative {
    f''4 d \stopStaff
    \override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
    \startStaff g, e |
    f'4 d \stopStaff
    \override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
    \startStaff g, e |
}\n
To preserve typical stem directions (in the bottom half of the staff stems point up, in the top half they point down), align the center line (or space) of the customized staff with the position of the normal center line (0). The clef position and the position of middle C may need to be adjusted accordingly to fit the new lines. See [Clef], page 17.

Staff line thickness can be altered. Ledger lines and note stems, by default, are also affected.

\new Staff \with {
  \override StaffSymbol.thickness = #3
} \relative {
  f''4 d g, e
}

It is also possible to set ledger line thickness independently of staff lines.

\new Staff \with {
  \override StaffSymbol.thickness = #2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f''''4 a, a,, f
}

The first value is multiplied by the staff line thickness, the second by the staff space and then the two values are added together to give the new thickness of the ledger line.

The vertical positions of ledger lines can be altered,

\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f''''4 a, a,, f
}

Additional ledger lines can be made to appear above or below note heads depending on the current position relative to other note heads that also have their own ledger lines.

\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
} \relative {
  f''''4 a, d, f,
Ledger lines can also be made to appear inside the staff where custom staff lines are required. The example shows the default position of ledger lines when the explicit \code{ledger-position} is and is not set. The \code{\stopStaff} is needed in the example to revert the \code{\override} for the whole StaffSymbol.

\begin{verbatim}
\relative d' {
    \override Staff.StaffSymbol.line-positions = #'(-8 0 2 4)
d4 e f g
\stopStaff
\startStaff
    \override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
d4 e f g
}
\end{verbatim}

The distance between staff lines can be altered. This affects ledger line spacing as well.

\begin{verbatim}
\new Staff \with {
    \override StaffSymbol.staff-space = #1.5
} \relative {
    f''4 d, g, e,
}
\end{verbatim}

Selected Snippets

\textit{Making some staff lines thicker than the others}

For educational purposes, a staff line can be thickened (e.g., the middle line, or to emphasize the line of the G clef). This can be achieved by adding extra lines very close to the line that should be emphasized, using the \code{line-positions} property of the StaffSymbol object.

\begin{verbatim}
{ 
    \override Staff.StaffSymbol.line-positions = 
        #'(-4 -2 -0.2 0 0.2 2 4)
d'4 e' f' g'
}
\end{verbatim}
See also

Music Glossary: Section “line” in Music Glossary, Section “ledger line” in Music Glossary, Section “staff” in Music Glossary.
Notation Reference: [Clef], page 17.
Snippets: Section “Staff notation” in Snippets.
Internals Reference: Section “StaffSymbol” in Internals Reference, Section “staff-symbol-interface” in Internals Reference.

Ossia staves

Ossia staves can be set by creating a new simultaneous staff in the appropriate location:

\new Staff \relative { 
  c''4 b d c 
  << 
    { c4 b d c } 
    \new Staff { e4 d f e } 
  >> 
  c4 b c2 
}

However, the above example is not what is usually desired. To create ossia staves that are above the original staff, have no time signature or clef, and have a smaller font size, tweaks must be used. The Learning Manual describes a specific technique to achieve this goal, beginning with Section “Nesting music expressions” in Learning Manual.

The following example uses the alignAboveContext property to align the ossia staff. This method is most appropriate when only a few ossia staves are needed.

\new Staff = "main" \relative { 
  c''4 b d c 
  << 
    { c4 b d c } 
    \new Staff \with { 
      \remove "Time_signature_engraver"
      alignAboveContext = "main"
      \magnifyStaff #2/3
      firstClef = &&f 
    } 
    { e4 d f e } 
  >> 
  c4 b c2 
}
If many isolated ossia staves are needed, creating an empty \texttt{Staff} context with a specific \texttt{context id} may be more appropriate; the ossia staves may then be created by \textit{calling} this context and using \texttt{\startStaff} and \texttt{\stopStaff} at the desired locations. The benefits of this method are more apparent if the piece is longer than the following example.

\begin{verbatim}
\new Staff = "ossia" \with { 
  \remove "Time_signature_engraver"
  \hide Clef
  \magnifyStaff #2/3
}
{ \stopStaff s1*6 }

\new Staff \relative { c'4 b c2
  <<
    { e4 f e2 }
  \context Staff = "ossia" { 
    \startStaff e4 g8 f e2 \stopStaff
  }
  >>
  g4 a g2 \break
  c4 b c2
  <<
    { g4 a g2 }
  \context Staff = "ossia" { 
    \startStaff g4 e8 f g2 \stopStaff
  }
  >>
  e4 d c2
}

Using the \texttt{\RemoveAllEmptyStaves} command to create ossia staves may be used as an alternative. This method is most convenient when ossia staves occur immediately following a line break. For more information about \texttt{\RemoveAllEmptyStaves}, see [Hiding staves], page 205.
Selected Snippets

Vertically aligning ossias and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossias.

\paper {
  ragged-right = ##t
}

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
{ \skip 2
  \new Staff \with {
    \lyrics {
      \set alignBelowContext = #'1
      lyrics4 below
    }
  }
}
alignAboveContext = #"3"
fontSize = #-2
\override StaffSymbol.staff-space = #(magstep -2)
\remove "Time_signature_engraver"

\tuplet 6/4 {
    \override TextScript.padding = #3
c8[^"ossia above" d e d e f]
}

\new Staff \with {
    \remove "Staff_symbol_engraver"
} \relative { a''8 f e16 d c b a2 }

See also
Notation Reference: [Hiding staves], page 205.
Snippets: Section “Staff notation” in Snippets.
Internals Reference: Section “StaffSymbol” in Internals Reference.

Hiding staves
Staff lines can be hidden by removing the Staff_symbol_engraver from the Staff context. As an alternative, \stopStaff may be used.
\new Staff \with {
    \remove "Staff_symbol_engraver"
} \relative { a''8 f e16 d c b a2 }

Empty staves can be hidden (for a so-called ‘Frenched Score’) by applying the \RemoveEmptyStaves command on a context, which can be done globally (in a \layout block)
as well as for specific staves only (in a `\with` block). This command removes all empty staves in a score except for those in the first system. If you want those in the first system to be hidden also, use `\RemoveAllEmptyStaves`.

```latex
\layout {
  \context {
    \Staff
    \RemoveEmptyStaves
  }
}

\relative <<
\new Staff {
  e'4 f g a \break
  b1 \break
  a4 b c2
}
\new Staff {
  c,4 d e f \break
  R1 \break
  f4 g c,2
}
>>
```

A staff is considered empty when it contains only multi-measure rests, rests, skips, or a combination of these elements. All other musical objects (that cause a staff not to be considered as empty) are listed in the `keepAliveInterfaces` context property, as initially set in the `ly/engraver-init.ly` file.

`\RemoveEmptyStaves` and `\RemoveAllEmptyStaves` are both predefined shortcuts that set such properties as `remove-empty` and `remove-first` for the `VerticalAxisGroup` object, as explained in Section A.20 [Context modification identifiers], page 821.

The `Keep_alive_together_engraver` allows groups of staves to only be removed together and not individually. By default, it is part of the `PianoStaff` context: a piano part will only be
hidden when both of its staves are empty. Similarly, a common engraving practice in orchestral scores is to remove empty groups of staves rather than individual staves; that can be achieved by adding the `Keep_alive_together_engraver` to the relevant staff-grouping context, as explained Section 5.1.4 [Modifying context plug-ins], page 605, (see [Grouping staves], page 193, for the context names).

```latex
\layout {
  \context {
    \StaffGroup \RemoveEmptyStaves \consists "Keep_alive_together_engraver"
  }
}
```

In the following example, staves devoted to wind instruments are removed in the second system; however, the double bass is not, because it is part of the larger group of fretted strings, which is playing.
The `Keep_alive_together_ engraver` internally uses the `remove-layer` property of a staff’s `VerticalAxisGroup` to decide whether to print it or not when it is considered empty. That property may also be set directly, in which case it acts as a priority index: values closest to zero take precedence over higher numbers, and thus staves whose `remove-layer` is higher will be masked in favor of staves of a lower number.

This is particularly useful for ‘divisi’ staves, where some individual parts (see Section 1.6.3 [Writing parts], page 210) occasionally need to be expanded to more than one staff. In the following example, two parts are routed to *three* staves; however, all three staves are never printed at the same time:

- in the first systems, only a single one of them is shown, as the `keepAliveInterfaces` property has been set to an empty list – therefore the other two staves are considered empty and thus hidden, regardless of what they may contain;
- when that property gets unset (and thus reverts to its default setting), it is no longer preventing the two other staves from being printed; however, as their `remove-layer` setting is lower than the single staff’s, these two staves are now printed in its place.

Such substitutions are applied not just to notes, chords and other musical events that occur immediately after the new setting, but to the whole system where it takes place.

```latex
\layout {
  short-indent = 2\cm
  indent = 3\cm
  \context {
    \Staff {
      keepAliveInterfaces = #'()
    }
  }
}

\set {ViolinI} {
  \repeat unfold 24 { d'4 }
  \once \unset Staff.keepAliveInterfaces
  <d' g''>2
  \repeat unfold 14 { d'4 }
  \bar "| ."
}

\set {ViolinII} {
  \repeat unfold 24 { g4 }
}
\repeat unfold 14 \{ g4 \}
\bar "|."

\new StaffGroup \with { \consists "Keep_alive_together_engraver" } \<<
\new Staff \with { 
  instrumentName = "Violins"
  shortInstrumentName = "V I & II"
  \override VerticalAxisGroup.remove-layer = 2
} \<< \violI \ \violII >>
\new Staff \with { 
  instrumentName = "Violin I"
  shortInstrumentName = "V I"
  \RemoveAllEmptyStaves
  \override VerticalAxisGroup.remove-layer = 1
} \violI
\new Staff \with { 
  instrumentName = "Violin II"
  shortInstrumentName = "V II"
  \RemoveAllEmptyStaves
  \override VerticalAxisGroup.remove-layer = 1
} \violII
\>>

\RemoveAllEmptyStaves can also be used to create ossia sections for a staff. For details, see [Ossia staves], page 202.
Predefined commands
\RemoveEmptyStaves, \RemoveAllEmptyStaves.

See also
Music Glossary: Section “Frenched staff” in Music Glossary.
Notation Reference: Section 5.1.5 [Changing context default settings], page 607, [Staff symbol], page 199, [Ossia staves], page 202, [Hidden notes], page 231, [Invisible rests], page 61, Section 5.4.7 [Visibility of objects], page 641, Section A.20 [Context modification identifiers], page 821, [Grouping staves], page 193, Section 5.1.4 [Modifying context plug-ins], page 605.

Installed Files: ly/engraver-init.ly.
Snippets: Section “Staff notation” in Snippets.
Internals Reference: Section “ChordNames” in Internals Reference, Section “FiguredBass” in Internals Reference, Section “Lyrics” in Internals Reference, Section “Staff” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference, Section “Staff_symbol_engraver” in Internals Reference, Section “Axis_group_engraver” in Internals Reference, Section “Keep_alive_together_engraver” in Internals Reference.

Known issues and warnings
Removing Staff_symbol_engraver also hides bar lines. If bar line visibility is forced, formatting errors may occur. In this case, use the following overrides instead of removing the engraver:
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t

For the Known issues and warnings associated with \RemoveEmptyStaves see Section 5.1.5 [Changing context default settings], page 607.

1.6.3 Writing parts
This section explains how to prepare parts for orchestral or ensemble music, which often requires to insert instrument names into the score. Methods to quote other voices and to format cue notes are also described, as well as a way to contract multiple consecutive empty measures in individual parts.

Additionally, a method for printing divisi staves, sometimes used in individual or desk parts, can be found in [Hiding staves], page 205.

Instrument names
Instrument names can be printed on the left side of staves in the Staff, PianoStaff, StaffGroup, GrandStaff and ChoirStaff contexts. The value of instrumentName is used for the first staff, and the value of shortInstrumentName is used for all succeeding staves.

\new Staff \with {
  instrumentName = "Violin "
  shortInstrumentName = "Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}

Violin \noteheads.s2 /dots.dot /dots.dot /flags.d4 /flags.u4 /noteheads.s2
\[\text{\textbackslash markup}\] can be used to create more complex instrument names:

\begin{verbatim}
\new Staff \with {
    instrumentName = \markup {
        \column { "Clarinetti"
            \line { "in B" \smaller \flat }
    }
}
}\relative {
c''4 c,16 d e f g2
}
\end{verbatim}

When two or more staff contexts are grouped together, the instrument names and short instrument names are centered by default. To center multi-line instrument names, \[\text{\textbackslash center-column}\] must be used:

\begin{verbatim}
\<<
\new Staff \with {
    instrumentName = "Flute"
}\relative {
    f''2 g4 f
}
\new Staff \with {
    instrumentName = \markup {
        \center-column { "Clarinet"
            \line { "in B" \smaller \flat }
        }
    }
}\relative { c''4 c,16 d e f g2 }
>>
\end{verbatim}

However, if the instrument names are longer, the instrument names in a staff group may not be centered unless the \texttt{indent} and \texttt{short-indent} settings are increased. For details about these settings, see [\texttt{paper} variables for shifts and indents], page 553.

\begin{verbatim}
\<<
\new Staff \with {
    instrumentName = "Alto Flute in G"
    shortInstrumentName = "Flt."
}\relative {
    f''2 g4 f \break
\end{verbatim}
To add instrument names to other contexts (such as ChordNames or FiguredBass), Instrument_name_ engraver must be added to that context. For details, see Section 5.1.4 [Modifying context plug-ins], page 605.

The shortInstrumentName may be changed in the middle of a piece, along with other settings as needed for the new instrument. However, only the first instance of instrumentName will be printed and subsequent changes will be ignored:

prepPiccolo = <<\markup \italic { muta in Piccolo }>

prepFlute = <<\markup \italic { muta in Flauto }>

setPiccolo = {
  \markup \bold { Piccolo }
  \transposition c''
}

setFlute = {
  \markup \bold { Flute }
  \transposition c'
}
\new Staff \with {
  instrumentName = "Flute"
  shortInstrumentName = "Flt."
} \relative {
  g'1 g g g \break
  g1 g \setPiccolo R R \break
  \set Staff.instrumentName = "Piccolo"
  \set Staff.shortInstrumentName = "Picc."
  \setPiccolo
  g1 g g \break
  g1 g \setFlute R R \break
  \set Staff.instrumentName = "Flute"
  \set Staff.shortInstrumentName = "Flt."
  \setFlute
  g1 g g
}

See also

Notation Reference: \[\textit{paper} variables for shifts and indents\], page 553, Section 5.1.4 [Modifying context plug-ins], page 605.

Snippets: Section “Staff notation” in \textit{Snippets}.

Internals Reference: Section “InstrumentName” in \textit{Internals Reference}, Section “PianoStaff” in \textit{Internals Reference}, Section “Staff” in \textit{Internals Reference}. 
Quoting other voices

It is very common for one voice to use the same notes as those from another voice. For example, first and second violins playing the same phrase during a particular passage of the music. This is done by letting one voice quote the other, without having to re-enter the music all over again for the second voice.

The `\addQuote` command, used in the top level scope, defines a stream of music from which fragments can be quoted.

The `\quoteDuring` command is used to indicate the point where the quotation begins. It is followed by two arguments: the name of the quoted voice, as defined with `\addQuote`, and a music expression for the duration of the quote.

```latex
\addQuote "flute" { \fluteNotes }
```

If the music expression used in `\quoteDuring` contains notes instead of spacer or multimeasure rests then the quote will appear as polyphony and may produce unexpected results.

```latex
\addQuote "flute" { \fluteNotes }
```

\score { <<
  \new Staff \with { instrumentName = "Flute" } \fluteNotes
  \new Staff \with { instrumentName = "Oboe" } \oboeNotes
>> }

If the music expression used in `\quoteDuring` contains notes instead of spacer or multimeasure rests then the quote will appear as polyphony and may produce unexpected results.
If an \unfoldRepeats command in a music expression is required to be printed when using \quoteDuring, then it too must also contain its own \unfoldRepeats command;

\begin{verbatim}
fluteNotes = \relative {
   \repeat volta 2 { a'4 gis g gis }
}

oboeNotesDW = \relative {
   \repeat volta 2 \quoteDuring "incorrect" { s1 }
}

oboeNotesW = \relative {
   \repeat volta 2 \quoteDuring "correct" { s1 }
}

\addQuote "incorrect" { \fluteNotes }

\addQuote "correct" { \unfoldRepeats \fluteNotes }
\end{verbatim}

\begin{verbatim}
\score {
   \unfoldRepeats
   \new Staff \with { instrumentName = "Flute" }
   \fluteNotes
   \new Staff \with { instrumentName = "Oboe (incorrect)" }
   \oboeNotesDW
   \new Staff \with { instrumentName = "Oboe (correct)" }
   \oboeNotesW
   >>
}
\end{verbatim}
The \quoteDuring command uses the \transposition settings of both quoted and quoting parts to produce notes for the quoting part that have the same sounding pitch as those in the quoted part.

```
clarinetNotes = \relative c'' { 
    \transposition bes 
    \key d \major 
    b4 ais a ais | cis4"quoted" r8 bis\p b4( f)
}
```

```
oboeNotes = \relative { 
    c''4 cis c b \quoteDuring "clarinet" { s1 }
}
```

```
\addQuote "clarinet" { \clarinetNotes }
```

```
\score {
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes 
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes 
}
```

By default quoted music will include all articulations, dynamics, markups, etc., in the quoted expression. It is possible to choose which of these objects from the quoted music are displayed by using the quotedEventTypes context property.

```
fluteNotes = \relative { 
    a'2 g2 |
    b4<"quoted" r8 ais a4\f( c->)
}
```

```
oboeNotes = \relative { 
    c''2. b4 |
    \quoteDuring "flute" { s1 }
}
```
\addQuote "flute" { \fluteNotes }

\score {
  \set Score.quotedEventTypes = #'(note-event articulation-event
crescendo-event rest-event
slur-event dynamic-event)

  \new Staff \with { instrumentName = "Flute" } \fluteNotes
  \new Staff \with { instrumentName = "Oboe" } \oboeNotes
}

Quotes can also be tagged, see [Using tags], page 520.

See also

Notation Reference: [Instrument transpositions], page 27, [Using tags], page 520.

Installed Files: scm/define-event-classes.scm.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “Music classes” in Internals Reference, Section “QuoteMusic” in Internals Reference, Section “Voice” in Internals Reference.

Known issues and warnings

Only the contents of the first Voice occurring in an \addQuote command will be considered for quotation, so if the music expression contains \new or \context Voice statements, their contents will not be quoted. Quoting grace notes is unsupported and may cause LilyPond to crash whereas quoting nested triplets may result in poor notation.

Formatting cue notes

The simplest way to format cue notes is to explicitly create a CueVoice context within the part.

\relative { 
  \new CueVoice { 
    \stemUp d'8"flute" c d e fis2 
  } 
  d,4 r a r
}
The \texttt{\textbackslash cueClef} command can also be used with an explicit \texttt{CueVoice} context if a change of clef is required and will print an appropriately sized clef for the cue notes. The \texttt{\textbackslash cueClefUnset} command can then be used to switch back to the original clef, again with an appropriately sized clef.

\begin{verbatim}
relative {
    \clef "bass"
    R1
    <<
      \{ e'2\rest r4. \textbackslash cueClefUnset e,8 \}
      \new CueVoice {
        \textbackslash cueClef "treble" \stemUp d''8"flute" c d e fis2
      }
    >>
    d,,4 r a r
}
\end{verbatim}

For more complex cue note placement, e.g. including transposition, or inserting cue notes from multiple music sources the \texttt{\textbackslash cueDuring} or \texttt{\textbackslash cueDuringWithClef} commands can be used. These are more specialized form of \texttt{\textbackslash quoteDuring}, see [Quoting other voices], page 214, in the previous section.

The syntax is:

\begin{verbatim}
\textbackslash cueDuring quotename #direction music
\end{verbatim}

and

\begin{verbatim}
\textbackslash cueDuringWithClef quotename #direction #clef music
\end{verbatim}

The music from the corresponding measures of the \textit{quote name} is added as a \texttt{CueVoice} context and occurs simultaneously with the \textit{music}, which then creates a polyphonic situation. The \textit{direction} takes the argument \texttt{UP} or \texttt{DOWN}, and corresponds to the first and second voices respectively, determining how the cue notes are printed in relation to the other voice.

\begin{verbatim}
fluteNotes = \relative {
    r2. c''4 | d8 c d e fis2 | g2 d |
}\end{verbatim}
It is possible to adjust which aspects of the music are quoted with \cueDuring by setting the quotedCueEventTypes property. Its default value is '(note-event rest-event tie-event beam-event tuplet-span-event)', which means that only notes, rests, ties, beams and tuplets are quoted, but not articulations, dynamic marks, markup, etc.

\begin{quote}
\textbf{Note:} When a \texttt{Voice} starts with \texttt{\cueDuring}, as in the following example, the \texttt{Voice} context must be explicitly declared, or else the entire music expression would belong to the \texttt{CueVoice} context.
\end{quote}

Markup can be used to show the name of the quoted instrument. If the cue notes require a change in clef, this can be done manually but the original clef should also be restored manually at the end of the cue notes.

\begin{verbatim}
fluteNotes = \relative {
    r2 r8 d''16(\f f e g f a)
g8 g16 g g2.
}
\addQuote "oboe" { \oboeNotes }
\end{verbatim}
r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
    \clef bass
    R1
    \clef treble
    <>\markup \tiny \{ flute \}
    \cueDuring "flute" #UP \{ R1 \}
    \clef bass
    g4. b8 d2
}
\addQuote "flute" { \fluteNotes }
\new Staff {
    \bassoonNotes
}

Alternatively, the \cueDuringWithClef function can be used instead. This command takes an extra argument to specify the change of clef that needs to be printed for the cue notes but will automatically print the original clef once the cue notes have finished.

fluteNotes = \relative {
    r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
    \clef bass
    R1
    <>\markup \tiny "flute"
    \cueDuringWithClef "flute" #UP "treble" \{ R1 \}
    g4. b8 d2
}
\addQuote "flute" { \fluteNotes }
\new Staff {
    \bassoonNotes
}

Like \quoteDuring, \cueDuring takes instrument transpositions into account. Cue notes are produced at the pitches that would be written for the instrument receiving the cue to produce the sounding pitches of the source instrument.
To transpose cue notes differently, use \texttt{\textbackslash transposedCueDuring}. This command takes an extra argument to specify (in absolute mode) the printed pitch that you want to represent the sound of a concert middle C. This is useful for taking cues from an instrument in a completely different register.

```
piccoloNotes = \relative { 
    \clef "treble\^8"
    R1
    c'''8 c c e g2
    c4 g g2
}
```

```
bassClarinetNotes = \relative c' { 
    \key d \major
    \transposition bes, 
    d4 r a r
    \transposedCueDuring "piccolo" #UP d { R1 }
    d4 r a r
}
```

```
\addQuote "piccolo" { \piccoloNotes }
```

```
<<
    \new Staff \piccoloNotes
    \new Staff \bassClarinetNotes
>>
```

The \texttt{\textbackslash killCues} command removes cue notes from a music expression, so the same music expression can be used to produce the instrument part with cues and the score. The \texttt{\textbackslash killCues} command removes only the notes and events that were quoted by \texttt{\textbackslash cueDuring}. Other markup associated with cues, such as clef changes and a label identifying the source instrument, can be tagged for selective inclusion in the score; see [Using tags], page 520.

```
fluteNotes = \relative { 
    r2. c''4 d8 c d e fis2 g2 d2
}
```

```
bassoonNotes = \relative c { 
    \clef bass
    R1
    \tag #'part { 
        \clef treble
        \markup \tiny { flute }
    }
    \cueDuring "flute" #UP { R1 }
    \tag #'part \clef bass
```
g4. b8 d2
}\addQuote "flute" \fluteNotes
\new Staff { \bassoonNotes
}\new StaffGroup <<
\new Staff { \fluteNotes
\new Staff { \removeWithTag #'part \killCues \bassoonNotes }
} >>

See also
Notation Reference: [Quoting other voices], page 214, [Instrument transpositions], page 27, [Instrument names], page 210, [Clef], page 17, [Musical cues], page 320, [Using tags], page 520.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “CueVoice” in Internals Reference, Section “Voice” in Internals Reference.

Known issues and warnings
Collisions can occur with rests, when using \cueDuring, between Voice and CueVoice contexts. When using \cueDuringWithClef or \transposedCueDuring the extra argument required for each case must come after the quote and the direction.

Compressing empty measures
By default, all measures are printed even if they are empty – which can happen if a rhythmic event (such as notes, rests or invisible rests) is so long as to span several measures. This behaviour can be changed by contracting all empty measures into a single one, as illustrated here (the second part of this example, with expanded measures, actually reverts back to the default behavior):
\override Score.BarNumber.break-visibility = ##(#f #t #t)
\time 2/4
Although that notation is syntactically correct, it may be confusing from a musical point of view, as illustrated in the previous example; hence the need for measure numbers to be explicitly printed, using the syntax described in \[Using break-visibility\], page 643.

Where such a notation may become more useful is when applied to \[Full measure rests\], page 62. A multi-measure rest will then be shown as a single measure containing a multi-measure rest symbol, with the number of measures of rest printed above the measure:

% Default behavior
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Rest measures contracted to single measure
\compressEmptyMeasures
r1 | R1*17 | R1*4 |
\expandEmptyMeasures
% Rest measures expanded again
\time 3/4
R2.*2 |

Unlike \compressEmptyMeasures, the music function \compressMMRests will only apply to rests, leaving any other events uncompressed. As a function rather than a property setting, its syntax differs slightly in that it must be followed by a music expression:

\compressMMRests {
  % Rests are compressed...
  R1*7
  % ... but notes can still span multiple measures.
  g'1 a'1*2 d'1
  R1*2
}

All of the commands described in this section actually rely on the skipBars internal property, which is set in the Score context as explained in Section 5.3.2 \[The set command\], page 620.
Predefined commands
\compressEmptyMeasures, \expandEmptyMeasures, \compressMMRests.

Selected Snippets

Numbering single measure rests
Multi measure rests show their length by a number except for single measures. This can be changed by setting \restNumberThreshold.

\{
  \compressEmptyMeasures
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 0
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 10
  R1 R1*10 R1*11
}\n
Changing form of multi-measure rests
If there are ten or fewer measures of rests, a series of longa and breve rests (called in German “Kirchenpausen” - church rests) is printed within the staff; otherwise a simple line is shown. This default number of ten may be changed by overriding the \expand-limit property.

\relative c'\ {
  \compressMMRests \{
    R1*2 | R1*5 | R1*9
    \override MultiMeasureRest.expand-limit = #3
    R1*2 | R1*5 | R1*9
  \}
}\n
See also
Notation Reference: [Using break-visibility], page 643, [Full measure rests], page 62, Section 5.3.2 [The set command], page 620.

Internals Reference: Section “MultiMeasureRest” in Internals Reference, Section “MultiMeasureRestNumber” in Internals Reference, Section “MultiMeasureRestScript” in Internals Reference, Section “MultiMeasureRestText” in Internals Reference.
1.7 Editorial annotations

This section discusses the various ways to change the appearance of notes and add analysis or educational emphasis.

1.7.1 Inside the staff

This section discusses how to add emphasis to elements that are inside the staff.

Selecting notation font size

![Musical notation example]

To change the size of the notation without changing the staff size, specify a magnification factor with the \magnifyMusic command:

\new Staff <<
\new Voice \relative {
\voiceOne
<e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
}
\new Voice \relative {
\voiceTwo
\magnifyMusic 0.63 {
\override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
r32 c' a c a c a c r c a c a c a c a c a c a c a c a c
}
}
>>

![Musical notation example with magnified notation]

The \override in the example above is a bug workaround. See the “Known issues and warnings” at the end of this section.
If a normal sized note head is merged with a smaller one, the size of the smaller note may need to be reset (with \once \normalsize) so that the stems and accidentals align properly:

\new Staff <<
\key fis \minor
\mergeDifferentlyDottedOn
\new Voice \relative {
 \voiceOne
 \magnifyMusic 0.63 {
 \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
 \once \normalsize cis'32( cis' gis b a fis \once \normalsize d d'
 \once \normalsize cis, cis' gis b a gis \once \normalsize fis fis'
 \once \normalsize fis, fis' ais, cis b gis \once \normalsize eis eis'
 \once \normalsize a, a' bis, d cis b \once \normalsize gis gis')
 }
}
\new Voice \relative {
 \voiceTwo
 cis'8. d16 cis8. fis16 fis8. eis16 a8. gis16
}
>>

The \magnifyMusic command is not intended for cue notes, grace notes, or ossia staves—there are more appropriate methods of entering each of those constructs. Instead, it is useful when the notation size changes in a single instrumental part on one staff, and where grace notes are not appropriate, such as in cadenza-like passages or in cases such as the above examples. Setting the \magnifyMusic value to 0.63 duplicates the dimensions of the CueVoice context.

\begin{Verbatim}
\textbf{Note:} The \magnifyMusic command should \textit{not} be used when also resizing the staff. See Section 4.2.2 [Setting the staff size], page 559.
\end{Verbatim}

\textbf{Resizing individual layout objects}

An individual layout object can be resized by using the \tweak or \override commands to adjust its font-size property:

\begin{verbatim}
\relative {
  % resize a note head
  <f' \tweak font-size -4 b e>-5
  % resize a fingering
  bes-\tweak font-size 0 -3
  % resize an accidental
  \once \override Accidental.font-size = -4 bes!~
  % resize an articulation
  \once \override Script.font-size = 4 bes!~
}\end{verbatim}
Understanding the \texttt{fontSize} property

The \texttt{fontSize} context property adjusts the relative size of all glyph-based notational elements in a context:

\begin{verbatim}
\relative {
  \time 3/4
  d''4--5 c8( b a g) |
  \set fontSize = -6
  e'4-- c!8-4( b a g) |
  \set fontSize = 0
  fis4---3 e8( d) fis4 |
  g2.
}
\end{verbatim}

The \texttt{fontSize} value is a number indicating the size relative to the standard size for the current staff height. The default \texttt{fontSize} is 0; adding 6 to any \texttt{fontSize} value doubles the printed size of the glyphs, and subtracting 6 halves the size. Each step increases the size by approximately 12%.

The scheme function \texttt{magnification->fontSize} is provided for convenience since the logarithmic units of the \texttt{fontSize} property are not entirely intuitive. For example, to adjust the musical notation to 75% of the default size, use:

\begin{verbatim}
\set fontSize = #(magnification->fontSize 0.75)
\end{verbatim}

The scheme function \texttt{magstep} does the opposite: it converts a \texttt{fontSize} value into a magnification factor.

The \texttt{fontSize} property will only affect notational elements that are drawn with glyphs, such as noteheads, accidentals, scripts, etc. It will not affect the size of the staff itself, nor will it scale stems, beams, or horizontal spacing. To scale stems, beams, and horizontal spacing along with the notation size (without changing the staff size), use the \texttt{magnifyMusic} command discussed above. To scale everything, including the staff size, see Section 4.2.2 [Setting the staff size], page 559.

Whenever the \texttt{fontSize} context property is set, its value is added to the value of the \texttt{fontSize} \texttt{grob} property for individual layout objects, before any glyphs are printed. This can cause confusion when setting individual \texttt{fontSize} properties while \texttt{fontSize} is already set:

\begin{verbatim}
% the default font-size for NoteHead is 0
% the default font-size for Fingering is -5
\c''4-3
\set fontSize = -3
% the effective font size for NoteHead is now -3
\end{verbatim}
The effective font size for Fingering is now -8
\overline{c''4-3}

\override Fingering.font-size = 0
The effective font size for Fingering is now -3
\overline{c''4-3}

\relative c'' {
  \teeny c4.\rightarrow d8---3
  \tiny c4.\rightarrow d8---3
  \small c4.\rightarrow d8---3
  \normalsize c4.\rightarrow d8---3
  \large c4.\rightarrow d8---3
  \huge c4.\rightarrow d8---3
}

Font size changes are achieved by scaling the design size that is closest to the desired size. The standard font size (for font-size = 0) depends on the standard staff height. For a 20pt staff, an 11pt font is selected.

**Predefined commands**
\magnifyMusic, \teeny, \tiny, \small, \normalsize, \large, \huge.

**See also**
Notation Reference: [Selecting font and font size], page 254, Section 4.2.2 [Setting the staff size], page 559, [Formatting cue notes], page 217, [Ossia staves], page 202.
Snippets: Section “Editorial annotations” in Snippets.
Internals Reference: Section “font-interface” in Internals Reference.
Known issues and warnings

There are currently two bugs that are preventing proper horizontal spacing when using \magnifyMusic. There is only one available workaround, and it is not guaranteed to work in every case. In the example below, replace the mag variable with your own value. You may also try removing one or both of the \newSpacingSection commands, and/or the \override and \revert commands:

\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [music]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}

Fingering instructions

Fingering instructions can be entered using 'note-digit':

\relative { c'4-1 d-2 f-4 e-3 }

Markup texts or strings may be used for finger changes.

\relative {
  c'4-1 d-2 f\finger \markup \tied-lyric "4~3" c\finger "2 - 3"
}

A thumb fingering can be added (e.g., cello music) to indicate that a note should be played with the thumb.

\relative { <a'_\thumb a'-3>2 <b_\thumb b'-3> }

Fingerings for chords can also be added to individual notes by adding them after the pitches.

\relative {
  <c''-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
}

Fingering instructions may be manually placed above or below the staff, see Section 5.4.2 [Direction and placement], page 634.
Selected Snippets

Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, it must be used within a chord construct $\langle\rangle$, even for single notes. Orientation for string numbers and right-hand fingerings may be set in a similar way.

\relative c' {
  \set fingeringOrientations = #'(left)
  \set stringNumberOrientations = #'(up left down)
  \set strokeFingerOrientations = #'(down right up)
  \relative c' {
    \set fingerings = \up \down \downrightup \down \right \left \left \left
    \set stringNumberOrientations = #'(up left down)
    \set strokeFingerOrientations = #'(down right up)
    \set stringNumberOrientations = #'(up left down)
    \set strokeFingerOrientations = #'(down right up)
  }
}

Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff; that behavior, however, may be disabled. Attention needs to be paid to situations where fingerings and stems are in the same direction: by default, fingerings will avoid only beamed stems. That setting can be changed to avoid no stems or all stems; the following example demonstrates these two options, as well as how to go back to the default behavior.

\relative c' {
  \override Fingering.staff-padding = #'() \\
  \override Fingering.add-stem-support = ##f \\
  a[-1 b]-2 \right c\rightHandFinger #1 e\rightHandFinger #2 c'\rightHandFinger #4 > 
}

\relative c' {
  \override Fingering.staff-padding = #'() \\
  \override Fingering.add-stem-support = ##t \\
  a[-1 b]-2 \right c\rightHandFinger #1 e\rightHandFinger #2 c'\rightHandFinger #4 > 
}
See also

Notation Reference: Section 5.4.2 [Direction and placement], page 634.
Snippets: Section “Editorial annotations” in Snippets.

Hidden notes
Hidden (or invisible or transparent) notes can be useful in preparing theory or composition exercises.

\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
  b
  \unHideNotes
  c
}\n
Note heads, stems, and flags, and rests are invisible. Beams are invisible if they start on a hidden note. Objects that are attached to invisible notes are still visible.

\relative c' {
  e8(\p f g a)\--
  \hideNotes
  e8(\p f g a)\--
}\n
Predefined commands
\hideNotes, \unHideNotes.

See also

Notation Reference: [Invisible rests], page 61, Section 5.4.7 [Visibility of objects], page 641, [Hiding staves], page 205.
Coloring objects

Individual objects may be assigned colors. Valid color names are listed in the Section A.7 [List of colors], page 681.

\override NoteHead.color = #red
c''4 c''
\override NoteHead.color = #(x11-color 'LimeGreen)
d''
\override Stem.color = "deepskyblue"
e''

\begin{staff}
\with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
  }
}
\relative c'' {\new Staff {c'' d'' e'' f'' g'8 \huge r4 a'16 f'}}
\end{staff}

In addition to a limited set of simple colors available as predefined variables (see ‘Normal colors’ in Section A.7 [List of colors], page 681), any color may be entered as a string. That string may be either a CSS (https://www.w3.org/Style/CSS)-style predefined color name, or a hexadecimal color code prefixed by the # character (inside the double quotes):

\override NoteHead.color = "lightsalmon"
\override Flag.color = "#E30074"
\override Beam.color = "#5e45ad"
\override Rest.color = "#3058"
g'8 \huge r4 a'16 f'

\begin{staff}
\relative c'' {\new Staff {c'' d'' e'' f'' g'8 \huge r4 a'16 f'}}
\end{staff}

If that color code includes an alpha channel for semi-transparency, by using an eight-character code "#RRGGBBAA" or its shorthand form "#RGBA", it will be used in SVG output but not in PostScript/PDF output. In the previous example, the rest can be seen through only if the code is compiled with the SVG backend, as explained in Section 3.4.3 [Alternative output formats], page 528.

In a different way, the full range of colors defined for X11 (https://en.wikipedia.org/wiki/X11_color_names) can be accessed by using the Scheme function x11-color. That function takes one argument, which can be a symbol, such as 'DarkSeaGreen4, or a string, such as "DarkSeaGreen4". The first form is quicker to write and slightly more efficient; however, the second form also makes it possible to specify X11 colors as multiple words: in this instance, "dark sea green 4".

If x11-color cannot make sense of the parameter, then the color returned defaults to black.

\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
  }
}
\relative c'' {\new Staff {c'' d'' e'' f'' g'8 \huge r4 a'16 f'}}
\override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
Exact RGB colors can be specified using the Scheme function `rgb-color`. This function takes three arguments used respectively for the red, green and blue channels, and an optional alpha number for semi-transparency. (All values must be numbers from 0 to 1.) Again, transparency is only supported in SVG output; for example, in the following fragment the staff’s clef can be seen through when rendered in SVG.

\new Staff \with {
  \instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
  }
  \override Clef.color = #(rgb-color 0 0 0 0.5)
}
\relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  \override Stem.color = #(rgb-color 0 0 0)
  gis8 a
  \override Stem.color = #(rgb-color 1 1 1)
  gis8 a
  \override Stem.color = #(rgb-color 0 0 0.5)
  gis4 a
}

See also

Notation Reference: Section A.7 [List of colors], page 681, Section 5.3.4 [The tweak command], page 623.

Snippets: Section “Editorial annotations” in *Snippets*.

Known issues and warnings

An X11 color is not necessarily exactly the same shade as a similarly named normal color.

Not all X11 colors are distinguishable in a web browser, i.e., a web browser might not display a difference between `LimeGreen` and `ForestGreen`. For web use CSS colors are recommended, as detailed in Section A.7 [List of colors], page 681.
Notes in a chord cannot be separately colored with \override; use \tweak or the equivalent \single\override before the respective note instead, see Section 5.3.4 [The tweak command], page 623.

**Parentheses**

Objects may be parenthesized by prefixing \parenthesize to the music event. When prefixed to a chord, it parenthesizes every note. Individual notes inside a chord may also be parenthesized.

\relative {
    c'\pw \parenthesize d
    c2 \parenthesize <c e g>
    c2 <c \parenthesize e g>
}

Non-note objects may be parenthesized as well. For articulations, a hyphen is needed before the \parenthesize command.

\relative {
    c'\pw-\parenthesize -. d
    c2 \parenthesize r
}

See also

Snippets: Section “Editorial annotations” in Snippets.

Internals Reference: Section “Parenthesis_ engraver” in Internals Reference, Section “ParenthesesItem” in Internals Reference, Section “parentheses-interface” in Internals Reference.

**Known issues and warnings**

Parenthesizing a chord prints parentheses around each individual note, instead of a single large parenthesis around the entire chord.

**Stems**

Whenever a note is found, a Stem object is created automatically. For whole notes and rests, they are also created but made invisible.

Stems may be manually placed to point up or down; see Section 5.4.2 [Direction and placement], page 634.

**Predefined commands**

\stemUp, \stemDown, \stemNeutral.
Selected Snippets

**Default direction of stems on the center line of the staff**

The default direction of stems on the center line of the staff is set by the `Stem` property `neutral-direction`.

```
\relative c'' {
    a4 b c b
    \override Stem.neutral-direction = #up
    a4 b c b
    \override Stem.neutral-direction = #down
    a4 b c b
}
```

![Staff notation example](https://example.com/staff_notation_example.png)

**Automatically changing the stem direction of the middle note based on the melody**

LilyPond can alter the stem direction of the middle note on a staff so that it follows the melody, by adding the `Melody_engraver` to the Voice context and overriding the `neutral-direction` of Stem.

```
\relative c'' {
    \time 3/4
    a8 b g f b g |
    c b d c b c |
}
```

```
\layout {
    \context {
        \Voice
        \consists "Melody_engraver"
        \autoBeamOff
        \override Stem.neutral-direction = #'()
    }
}
```

![Staff notation example](https://example.com/staff_notation_example.png)

See also

Notation Reference: Section 5.4.2 [Direction and placement], page 634.

Snippets: Section “Editorial annotations” in Snippets.

Internals Reference: Section “Stem_engraver” in Internals Reference, Section “Stem” in Internals Reference, Section “stem-interface” in Internals Reference.

**1.7.2 Outside the staff**

This section discusses how to add emphasis to elements in the staff from outside of the staff.
Note names

Note names can be printed as text, by using the \NoteNames context. When used simultaneously with a regular staff, that makes it possible to synchronize each note with its name, printed above or below the Staff.

```lilypond
\language "italiano"
melody = \relative do' {
    fad2 si,8 dod re mi fad4. re8 fad2
}
```

By default, note names are printed in the same language used for music entry; however, the \printNotesLanguage property allows to select any other language available (see [Note names in other languages], page 8). Whether accidentals should be printed or not is determined through the \printAccidentalNames property.

By setting both that property to a symbol and \printOctaveNames to ##t, note names can be obtained that closely resemble LilyPond entry syntax. If a more generalistic result is desired, 'scientific' octave names may also be obtained.

```lilypond
melody = \relative c' {
    fis2 b,8 cis d e fis4. d8 fis2
}
```

By default, note names are printed in the same language used for music entry; however, the \printNotesLanguage property allows to select any other language available (see [Note names in other languages], page 8). Whether accidentals should be printed or not is determined through the \printAccidentalNames property.

By setting both that property to a symbol and \printOctaveNames to ##t, note names can be obtained that closely resemble LilyPond entry syntax. If a more generalistic result is desired, 'scientific' octave names may also be obtained.
The `noteNameSeparator` property defines how chords will be printed. Other formatting functions may be defined as `noteNameFunction`; such a function must expect a `pitch` and a `context` argument, even if one of these can then be ignored.

```lilypond
somechords = \relative c' {
    <b d fis>2 <b cis e g> <b d fis> q
}

<<
\new NoteNames {
    \set noteNameSeparator = "+
    \somechords
}
\new Staff { \key b \minor \somechords }
\new NoteNames {
    \set noteNameFunction =
        #(lambda (pitch ctx)
            (alteration->text-accidental-markup
                (ly:pitch-alteration pitch)))
    \somechords
}
>>
```

See also

Notation Reference: [Note names in other languages], page 8.

Internals Reference: Section “NoteName” in Internals Reference, Section “NoteNames” in Internals Reference, Section “Note_name_engraver” in Internals Reference.

Balloon help

Elements of notation can be marked and named with the help of a square balloon. The primary purpose of this feature is to explain notation.

```lilypond
\new Voice \with { \consists "Balloon_engraver" }
\relative c' {
    \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" }
    a8
    \balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" }
    r
    <c, g'-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2.
}
```
Chapter 1: Musical notation

There are two music functions, \(\text{balloonGrobText}\) and \(\text{balloonText}\); the former is used like \(\text{\textbackslash once \textbackslash override}\) to attach text to any grob, and the latter is used like \(\text{\textbackslash tweak}\), typically within chords, to attach text to an individual note.

Balloon text does not influence note spacing, but this can be altered:

\[
\begin{align*}
\text{\textbackslash new Voice} & \text{ \with { \consists "Balloon_engraver" } } \\
\text{\relative c''} & \{ \\
\text{\textbackslash balloonGrobText} & \#\text{Stem} \ '#(3.4) \text{ \markup { "I'm a Stem" } a8} \\
\text{\textbackslash balloonGrobText} & \#\text{Rest} \ '#(-4.-4) \text{ \markup { "I'm a rest" } r} \\
\text{\textbackslash balloonLengthOn} & <c, g'\text{\textbackslash balloonText} '#(-2.-2) \text{ \markup { "I'm a note head" } c>2.} \\
\end{align*}
\]

Predefined commands

\(\text{\textbackslash balloonLengthOn}, \text{\textbackslash balloonLengthOff}.\)

See also

Snippets: Section “Editorial annotations” in \textit{Snippets}.


Grid lines

Vertical lines can be drawn between staves synchronized with the notes.

The \textit{Grid\_point\_engraver} must be used to create the end points of the lines, while the \textit{Grid\_line\_span\_engraver} must be used to actually draw the lines. By default this centers grid lines horizontally below and to the left side of each note head. Grid lines extend from the middle lines of each staff. The \textit{gridInterval} must specify the duration between the grid lines.

\[
\begin{align*}
\text{\textbackslash layout} & \{ \\
\text{\textbackslash context} & \{ \\
\text{\textbackslash Staff} & \{ \\
\text{\consists "Grid\_point\_engraver"} & \\
\text{\gridInterval} & = \#(\text{ly:make-moment} 1/4) \\
\} \\
\text{\textbackslash context} & \{ \\
\text{\textbackslash Score} & \{ \\
\text{\consists "Grid\_line\_span\_engraver"} & \\
\} \\
\end{align*}
\]
Selected Snippets

Grid lines: changing their appearance

The appearance of grid lines can be changed by overriding some of their properties.

\score {
\new ChoirStaff <<
\new Staff \relative { \stemUp c''4. d8 e8 f g4 }
\new Staff \relative { \clef bass \stemDown c4 g' f e }
>>
}

\score {
\new ChoirStaff <<
\new Staff { \relative c' { \stemUp c'4. d8 e8 f g4 }
}
\new Staff { \relative c { \% this moves them up one staff space from the default position
\override Score.GridLine.extra-offset = #'(0.0 . 1.0) \stemDown \clef bass \once \override Score.GridLine.thickness = #5.0 c4 \once \override Score/GridLine/thickness = #1.0 g'4 \once \override Score/GridLine/thickness = #3.0 f4 \once \override Score/GridLine/thickness = #5.0 e4 }
}
See also

Snippets: Section “Editorial annotations” in Snippets.

Internals Reference: Section “Grid_line_span_engraver” in Internals Reference, Section “Grid_point_engraver” in Internals Reference, Section “GridLine” in Internals Reference, Section “GridPoint” in Internals Reference, Section “grid-line-interface” in Internals Reference, Section “grid-point-interface” in Internals Reference.

Analysis brackets

Brackets are used in musical analysis to indicate structure in musical pieces. Simple horizontal brackets are supported.

\layout {
  \context {
    \Voice
      \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}
Analysis brackets may be nested.

\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative { c'4\startGroup\startGroup d4\stopGroup\stopGroup e4\startGroup d4\stopGroup\stopGroup}

\begin{music}
\relative c''4\startGroup\startGroup d4\stopGroup\stopGroup e4\startGroup\startGroup d4\stopGroup\stopGroup
\end{music}

Selected Snippets

**Analysis brackets above the staff**

Simple horizontal analysis brackets are added below the staff by default. The following example shows a way to place them above the staff instead.

\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' { 
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup d2\stopGroup
}

\begin{music}
\relative c''4\startGroup\startGroup c2\stopGroup\stopGroup d2\stopGroup\stopGroup
\end{music}

**Analysis brackets with labels**

Text may be added to analysis brackets through the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command. Bracket text will be parenthesized after a line break.

\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}
\once\override HorizontalBracketText.text = "a"
c''\startGroup d''\stopGroup
\once\override HorizontalBracketText.text = "a''"
e''\startGroup d''\stopGroup
c''
\tweak HorizontalBracketText.text \markup \bold \huge "b" \startGroup
d''\stopGroup
e''\tweak HorizontalBracketText.text "a''" \startGroup
d''\stopGroup\stopGroup
c''\tweak HorizontalBracketText.text foo \startGroup d' e' f'
\break
g'' a' b' c'\stopGroup

\begin{music}
\begin{Staff}
\bar{4}
\underline{\note{b}}
\note{a} \note{a'} \underline{\text{a}} \underline{\text{a'}} \underline{\text{foo}}
\note{\text{(foo)}}
\end{Staff}
\end{music}

See also


1.8 Text
This section explains how to include text (with various formatting) in music scores.

1.8.1 Writing text
This section introduces different ways of adding text to a score.

Note: To write accented and special text (such as characters from other languages), simply insert the characters directly into the LilyPond file. The file must be saved as UTF-8. For more information, see [Text encoding], page 525.

Text objects overview
Text objects are entered either as simple strings between double quotes, or as \markup blocks that can accept a variety of advanced text formatting and graphical enhancements, as detailed in Section 1.8.2 [Formatting text], page 252.

As such, markup blocks may be used:
- in any TextScript object (attached to notes with -, ^ or _); see [Text scripts], page 245;
- as ‘spanners’, when some indications are prolonged over several beats or bars; see [Text spanners], page 246;
- in any mark printed above the score, such as RehearsalMark or MetronomeMark objects respectively introduced with the \mark or \tempo keywords; see [Text marks], page 248;
- as standalone markup blocks, entered at the top level outside of any \score block (in this specific case the \markup {...} command is mandatory, and cannot be omitted in favor of a simple text string between double quotes); see [Separate text], page 251;
- in any definition inside the \header block (e.g., title, subtitle, composer), or in specific elements defined inside the \paper block such as evenHeaderMarkup for page numbers. This is explained in Section 3.2 [Titles and headers], page 492.

Many other text-based objects may be entered as markup blocks, even if that is not their primary use.
- Fingerings may easily be replaced with markup blocks, if introduced with the \finger command; see [Fingering instructions], page 229.
• Lyric syllables may be formatted through the \markup command; see Section 2.1.1 [Common notation for vocal music], page 274.

• Chord names are in fact defined as markup blocks, and therefore may be redefined in the same way for customizing chord modifiers or chord exceptions; see Section 2.7.2 [Displaying chords], page 429.

• Dynamics are usually entered in a simple way; however it is possible to define [New dynamic marks], page 131, as markup objects. Some dynamics such as crescendo are printed as spanners and may be redefined through properties such as crescendoText; see [Dynamics], page 125.

• Less common objects are also made of markup blocks, such as [Balloon help], page 237, indications.

   In fact, it is possible to use \markup to customize the appearance of virtually any graphical object (or ‘grob’), by overriding either its text property if it has one, or its stencil property. Some of the logic that makes this a possibility is explained in Section “Flexible architecture” in Essay.

   The following example illustrates the ubiquity of markup blocks, not only as some of the objects listed above, but also by replacing musical objects with text objects through various methods.

   \header { title = \markup "Header" }

   dyn =
   #(make-dynamic-script #{ \markup \text "DynamicText" #})

   \markup \box "Top-level markup"

   \score {
      <<
         \new ChordNames
         \with { majorSevenSymbol = \markup "majorSevenSymbol" }
         \chordmode { c1:maj7 }
         \new Staff {
            \tempo \markup "MetronomeMark"
            \mark \markup \smaller "RehearsalMark"
            \once \override TupletNumber.text = \markup "TupletNumber"
            \tuplet 3/2 {
               \once \override NoteHead.stencil = \\
                  #ly:text-interface::print
               \once \override NoteHead.text = \\
                  \markup \lower #0.5 "NoteHead"
               c'8\markup \italic "TextScript"
               a’\finger \markup \text "Fingering"
               \once \override Rest.stencil = \\
                  #(lambda (grob)
                     (grob-interpret-markup grob #{
                        \markup "Rest"
                        #}))
               r
            }
         }
      >>}
   }
Header

Top-level markup

RehearsalMark
MetronomeMark
C majorSevenSymbol
TextScript
Fingering
Rest
TupletNumber
LYRICTEXT
DynamicText

See also

Notation Reference: Section 1.8.2 [Formatting text], page 252, [Text scripts], page 245, [Text spanners], page 246, [Text marks], page 248, [Separate text], page 251, [Fingering instructions], page 229, Section 2.1.1 [Common notation for vocal music], page 274, Section 2.7.2 [Displaying chords], page 429, [New dynamic marks], page 131, [Dynamics], page 125, [Balloon help], page 237.

Essay on automated music engraving: Section “Flexible architecture” in Essay.

Snippets: Section “Text” in Snippets.

Text scripts

Simple “quoted text” indications may be added to a score, as demonstrated in the following example. Such indications may be manually placed above or below the staff, using the syntax described in Section 5.4.2 [Direction and placement], page 634.

\relative { a'8"pizz." g f e a4="scherz." f }

This syntax is actually a shorthand; more complex text formatting may be added to a note by explicitly using a \markup block, as described in Section 1.8.2 [Formatting text], page 252.

\relative {
  a'8\markup { \italic pizz. } g f e
  a4\markup { \tiny scherz. \bold molto } f }

\new Lyrics \lyricmode {
  \markup \smallCaps "LyricText" 1
}
\new Dynamics { s1\dyn }
}
By default, text indications do not influence the note spacing. However, their widths can be taken into account: in the following example, the first text string does not affect spacing, whereas the second one does.

\relative {
  a'8^"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}

In addition to text scripts, articulations can be attached to notes. For more information, see [Articulations and ornamentations], page 122.

For more information about the relative ordering of text scripts and articulations, see Section “Placement of objects” in Learning Manual.

**Predefined commands**

\textLengthOn, \textLengthOff.

**See also**


Notation Reference: Section 1.8.2 [Formatting text], page 252, Section 5.4.2 [Direction and placement], page 634, [Articulations and ornamentations], page 122.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

**Known issues and warnings**

Checking to make sure that text scripts and lyrics are within the margins requires additional calculations. In cases where slightly faster performance is desired, use

\override Score.PaperColumn.keep-inside-line = ##f

**Text spanners**

Some performance indications, e.g., rallentando or accelerando, are written as text and are extended over multiple notes with dotted lines. Such objects, called “spanners”, may be created from one note to another using the following syntax:

\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}

\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
Chapter 1: Musical notation

The string to be printed is set through object properties. By default it is printed in italic characters, but different formatting can be obtained using \markup blocks, as described in Section 1.8.2 [Formatting text], page 252.

\relative {  
  \override TextSpanner.bound-details.left.text =  
  \markup { \upright "rit." }  
  b'1\startTextSpan c  
  e,\stopTextSpan  
}\noteheads.s0\noteheads.s0\noteheads.s0  
\relative c' {  
c4\cresc d4 e4 f4 |  
g4 a4\! b4\crpoco c4 |  
c4 d4 e4 f4 |  
g4 a4\! b4\!< c4 |  
g4\dim a4 b4\decresc c4\!  
}\noteheads.s2\noteheads.s2\noteheads.s2\noteheads.s2\noteheads.s2  
cresc. - - - -  
cresc. poco a poco -< dim. decresc.

The line style, as well as the text string, can be defined as an object property. This syntax is described in Section 5.4.8 [Line styles], page 648.

Predefined commands
\textSpannerUp, \textSpannerDown, \textSpannerNeutral.

Known issues and warnings
LilyPond is only able to handle one text spanner per voice.

Selected Snippets

Dynamics text spanner postfix

Custom text spanners can be defined and used with hairpin and text crescendos. < and > produce hairpins by default, \cresc etc. produce text spanners by default.

% Some sample text dynamic spanners, to be used as postfix operators  
crpoco =  
#(make-music 'CrescendoEvent  
'\span-direction START  
'\span-type 'text  
'\span-text "cresc. poco a poco")

\relative c' {  
c4\cresc d4 e4 f4 |  
g4 a4\! b4\crpoco c4 |  
c4 d4 e4 f4 |  
g4 a4\! b4\!< c4 |  
g4\dim a4 b4\decresc c4\!  
}\relative c' {  
c4\cresc d4 e4 f4 |  
g4 a4\! b4\crpoco c4 |  
c4 d4 e4 f4 |  
g4 a4\! b4\!< c4 |  
g4\dim a4 b4\decresc c4\!  
}\noteheads.s2\noteheads.s2\noteheads.s2\noteheads.s2\noteheads.s2  
cresc. - - - -  
cresc. poco a poco -< dim. decresc.

Dynamics custom text spanner postfix

Postfix functions for custom crescendo text spanners. The spanners should start on the first note of the measure. One has to use -\mycresc, otherwise the spanner start will rather be assigned to the next note.

% Two functions for (de)crescendo spanners where you can explicitly give the
% spanner text.
mycresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'CrescendoEvent
   'span-direction START
   'span-type 'text
   'span-text mymarkup))
mydecresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'DecrescendoEvent
   'span-direction START
   'span-type 'text
   'span-text mymarkup))
\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4  c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4  c4\! c4 c4
}

\relative c' {
  \mark "Verse" c'2 g'
  \bar "||"
  \mark "Chorus" g2 c,
  \bar "."
}

Text marks
Various text elements may be added to a score using the syntax described in [Rehearsal marks], page 113:

\relative {
  \mark "Verse"
  c'2 g'
  \bar "||"
  \mark "Chorus"
  g2 c,
  \bar "."
}

This syntax makes it possible to put any text on a bar line; more complex text formatting may be added using a \markup block, as described in Section 1.8.2 [Formatting text], page 252:
\relative {

See also
Notation Reference: Section 5.4.8 [Line styles], page 648, [Dynamics], page 125, Section 1.8.2 [Formatting text], page 252.
Snippets: Section “Text” in Snippets, Section “Expressive marks” in Snippets.
Internals Reference: Section “TextSpanner” in Internals Reference.
This syntax also allows to print special signs, like coda, segno or fermata, by specifying the appropriate symbol name as explained in [Music notation inside markup], page 264:

\relative {
    <bes' f>2 <aes d>
    \mark \markup { \musicglyph "scripts.ufermata" }
    <e g>1
}

Such objects are only typeset above the top staff of the score; depending on whether they are specified at the end or the middle of a bar, they can be placed above the bar line or between notes. When specified at a line break, the mark will be printed at the beginning of the next line.

\relative c'' {
    \mark "Allegro"
    c1 c
    \mark "assai" \break
    c c
}

Predefined commands
\markLengthOn, \markLengthOff.

Selected Snippets

Printing marks at the end of a line

Marks can be printed at the end of the current line, instead of the beginning of the following line. In such cases, it might be preferable to align the right end of the mark with the bar line.

\relative c'' {

Printing marks on every staff

Although text marks are normally only printed above the topmost staff, they may also be printed on every staff.

\score { 
<< 
\new Staff { c''1 \mark "molto" c'' } 
\new Staff { c'1 \mark "molto" c' } 
>> 
\layout { 
\context { 
\Score 
\remove "Mark_engraver" 
\remove "Staff_collecting_engraver" 
} 
\context { 
\Staff 
\consists "Mark_engraver" 
\consists "Staff_collecting_engraver" 
} 
} 
}
A \markup block can exist by itself, outside of any \score block, as a “top-level expression”. This syntax is described in Section 3.1.5 [File structure], page 490.

\markup {
  Tomorrow, and tomorrow, and tomorrow...
}

\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}

\timesig.C44/clefs.G/noteheads.s0

Tomorrow, and tomorrow, and tomorrow...

Separate text blocks can be spread over multiple pages, making it possible to print text documents or books entirely within LilyPond. This feature, and the specific syntax it requires, are described in [Multi-page markup], page 267.

Predefined commands
\markup, \markuplist.
Selected Snippets

**Stand-alone two-column markup**

Stand-alone text may be arranged in several columns using \markup commands:

```lilypond
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column {\italic {
      \line { O sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }}
    \hspace #1
  }
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.

See also

Notation Reference: Section 1.8.2 [Formatting text], page 252, Section 3.1.5 [File structure], page 490, Section 3.1.2 [Multiple scores in a book], page 487, [Multi-page markup], page 267.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

1.8.2 Formatting text

This section presents basic and advanced text formatting, using the \markup mode specific syntax.

**Text markup introduction**

A \markup block is used to typeset text with an extensible syntax called “markup mode”. Such blocks may be used anywhere, as explained in [Text objects overview], page 243.

The markup syntax is similar to LilyPond’s usual syntax: a \markup expression is enclosed in curly braces {...}. A single word is regarded as a minimal expression, and therefore does not need to be enclosed with braces.
Unlike simple “quoted text” indications, \markup blocks may contain nested expressions or mark up commands, entered using the backslash \ character. Such commands only affect the first following expression.

\relative { 
  a'1-\markup intenso 
  a2 ^\markup { poco \italic più forte } 
  c e1 
  d2 _\markup { \italic "string. assai" } 
  e 
  b1 ^\markup { \bold { molto \italic agitato } } 
  c 
}

A \markup block may also contain quoted text strings. Such strings are treated as minimal text expressions, and therefore any markup command or special character (such as \ and #) will be printed verbatim without affecting the formatting of the text. Double quotation marks themselves may be printed by preceding them with backslashes.

\relative { 
  a'1 ^"\italic markup..." 
  a _\markup { \italic "... prints "italic" letters!" } 
  a a 
}

To be treated as a distinct expression, a list of words needs to be enclosed with double quotes or preceded by a command. The way markup expressions are defined affects how these expressions will be stacked, centered and aligned; in the following example, the second \markup expression is treated the same as the first one:

\relative c' { 
  c1 \markup { \center-column { a bbb c } } 
  c1 \markup { \center-column { a bbb c } } 
  c1 \markup { \center-column { a \line { bbb c } } } 
  c1 \markup { \center-column { a "bbb c" } } 
}

Markups can be stored in variables. Such variables may be directly attached to notes:

\set allegro = \markup { \bold \large Allegro }
An exhaustive list of \markup-specific commands can be found in Section A.11 [Text markup commands], page 711. The inner workings of these commands, and how to implement new ones, is explained in Section “Markup functions” in Extending.

See also
Notation Reference: [Text objects overview], page 243, Section A.11 [Text markup commands], page 711.
Extending LilyPond: Section “Markup functions” in Extending.
Snippets: Section “Text” in Snippets.
Installed Files: scm/markup.scm.

Known issues and warnings
Syntax error messages for markup mode can be confusing.

Selecting font and font size
Basic font switching is supported in markup mode:
\relative {
    d'1\markup {
        \bold { Più mosso }
        \italic { non troppo \underline Vivo }
    }
    r2 r4 r8
    d,\markup { \italic quasi \smallCaps Tromba }
    f1 d2 r
}

The font size can be altered, relative to the global staff size, in a number of different ways.
It can be set to predefined size.
\relative b' {
    b1\markup { \huge Sinfonia }
    b1\markup { \teeny da }
    b1-\markup { \normalsize camera }
}
It can be set relative to its previous value.
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6 camera }
}

It can be increased or decreased relative to the value set by the global staff size.
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}

It can also be set to a fixed point-size, regardless of the global staff size.
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}

If the text includes spaces, then it is best to put it all inside quote marks, so that the size of each space is appropriate for the size of the other characters.
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }

\section*{Sinfonia da camera}

Text may be printed as subscript or superscript. By default these are printed in a smaller size, but a normal size can be used as well:
\markup {
  \column {
    \line { 1 \super st movement }
  }
}
The markup mode provides an easy way to select alternate font families. The default serif font, of roman type, is automatically selected unless specified otherwise; on the last line of the following example, there is no difference between the first and the second word.

\markup {
  \column {
    \line { Act \number 1 } 
    \line { \sans { Scene I. } } 
    \line { \typewriter { Verona. An open place. } } 
    \line { Enter \roman Valentine and Proteus. }
  }
}

Act I
Scene I.
Verona. An open place.
Enter Valentine and Proteus.

Some of these font families, used for specific items such as numbers or dynamics, do not provide all characters, as mentioned in [New dynamic marks], page 131, and [Manual repeat marks], page 159.

When used inside a word, some font-switching or formatting commands may produce an unwanted blank space. This can easily be solved by concatenating the text elements together:

\markup {
  \column {
    \line { \concat { 1 \super st } movement }
    \line { \concat { \dynamic p , } \italic { con dolce espressione } }
  }
}

p, con dolce espressione

An exhaustive list of font switching commands and custom font usage commands can be found in Section A.11.1 [Font], page 711.

Defining custom font sets is also possible, as explained in Section 1.8.3 [Fonts], page 267.
Predefined commands
\teeny, \tiny, \small, \normalsize, \large, \huge, \smaller, \larger.

See also
Notation Reference: Section A.11.1 [Font], page 711, [New dynamic marks], page 131, [Manual repeat marks], page 159, Section 1.8.3 [Fonts], page 267.
Installed Files: scm/define-markup-commands.scm.
Snippets: Section “Text” in Snippets.
Internals Reference: Section “TextScript” in Internals Reference.

Known issues and warnings
Using the font sizing commands \teeny, \tiny, \small, \normalsize, \large, and \huge will lead to inconsistent line spacing compared to using \fontsize.

Text alignment
This subsection discusses how to place text in markup mode. Markup objects can also be moved as a whole, using the syntax described in Section “Moving objects” in Learning Manual.

Markup objects may be aligned in different ways. By default, a text indication is aligned on its left edge: in the following example, there is no difference between the first and the second markup. That example also demonstrates various syntactically correct ways of placing the alignment commands:
\relative {
   d'1-\markup { poco } f 
d-\markup { \left-align poco } f 
d-\markup { \center-align { poco } } f 
d-\markup \right-align { poco } 
}

Horizontal alignment may be fine-tuned using a numeric value:
\relative {
   a'1-\markup { \halign #-1 poco } e' 
a,-\markup { \halign #0 poco } e' 
a,-\markup { \halign #0.5 poco } e' 
a,-\markup { \halign #2 poco } 
}

Known issues and warnings
Using the font sizing commands \teeny, \tiny, \small, \normalsize, \large, and \huge will lead to inconsistent line spacing compared to using \fontsize.

Text alignment
This subsection discusses how to place text in markup mode. Markup objects can also be moved as a whole, using the syntax described in Section “Moving objects” in Learning Manual.

Markup objects may be aligned in different ways. By default, a text indication is aligned on its left edge: in the following example, there is no difference between the first and the second markup. That example also demonstrates various syntactically correct ways of placing the alignment commands:
\relative {
   d'1-\markup { poco } f 
d-\markup { \left-align poco } f 
d-\markup { \center-align { poco } } f 
d-\markup \right-align { poco } 
}

Horizontal alignment may be fine-tuned using a numeric value:
\relative {
   a'1-\markup { \halign #-1 poco } e' 
a,-\markup { \halign #0 poco } e' 
a,-\markup { \halign #0.5 poco } e' 
a,-\markup { \halign #2 poco } 
}
Lastly, words and any other objects may be moved by preceding them with padding. Negative padding is also supported, and will move any objects that follow in the opposite direction. Although padding is normally invisible, in the following example some commands have been added to make it appear more clearly:

\begin{code}{\relative d'\markup{ poco}}
\begin{code}{\relative d\markup{ \with-color #darkred \box \hspace #4 poco}}
\begin{code}{\relative d\markup{ \with-color #darkred \box \hspace #-4 poco}}
\begin{code}{\relative d\markup{ \with-color #darkred \box \hspace #10 poco}}
\end{code}
\end{code}
\end{code}
\end{code}

\begin{code}{\relative d'2\markup{ Acte I \raise #2 \Scène 1}}
\begin{code}{\relative a' \markup{ g \lower #4 \bold \Très modéré}}
\begin{code}{\relative a \markup{ d \raise #4 \italic Une forêt.}}
\begin{code}{\relative a'4 a g2 a}
\end{code}
\end{code}
\end{code}
\end{code}

Some commands can affect both the horizontal and vertical alignment of text objects in markup mode:

\begin{code}{\relative d'2\markup{ Acte I}}
\begin{code}{\relative a' \markup{ g \lower #4 \bold \Très modéré}}
\begin{code}{\relative a \markup{ d \raise #4 \italic Une forêt.}}
\begin{code}{\relative a'4 a g2 a}
\end{code}
\end{code}
\end{code}
Here again, padding (either positive or negative) is a convenient way of positioning objects vertically, when inserted into markup columns:

```latex
\relative {
  d'2 \markup {  
    Acte I  
    \column {  
      \with-color #darkred \box \vspace #-1  
      "Scène 1"  
    }  
  }  
  a'  
  g_ \markup {  
    \general-align #Y #3.2 \bold "Très modéré"  
  }  
  a  
  d, \markup \translate-scaled #'(-1 . 2) \teeny {  
    "Une forêt."  
  }  
  a'4 a g2 a  
}
```

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Example of musical notation with text.}
\end{figure}

A markup object may include several lines of text. In the following example, each element or expression is placed on its own line, either left-aligned or centered:

```latex
\relative {
  d'2 \markup {  
    Acte I  
    \column {  
      \with-color #darkred \box \vspace #1  
      "Scène 1"  
    }  
  }  
  a'  
  g_ \markup {  
    \general-align #Y #3.2 \bold "Très modéré"  
  }  
  a  
  d, \markup \translate-scaled #'(-1 . 2) \teeny {  
    "Une forêt."  
  }  
  a'4 a g2 a  
}
```

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Example of musical notation with text.}
\end{figure}
Similarly, a list of elements or expressions may be spread to fill the entire horizontal line width (if there is only one element, it will be centered on the page). These expressions can, in turn, include multi-line text or any other markup expression:

\markup { 
\fill-line { 
  \line { \text{William S. Gilbert} 
  \center-column { 
    \huge \text{smallCaps "The Mikado"} 
    or 
    \text{smallCaps "The Town of Titipu"} 
  } 
  \line { \text{Sir Arthur Sullivan} } 
} 
} 
\markup { 
  \fill-line { 1885 } 
}

William S. Gilbert \hspace{0.5cm} THE MIKADO \hspace{0.5cm} Sir Arthur Sullivan 

or

THE TOWN OF TITIPU

1885

Elements may be spread to fill any specified width by overriding the line-width property. By default it is set to #f which indicates the entire line:

\markup { 
\column { 
  \fill-line { left center right } 
  \null 
  \override #'(line-width . 30) 
  \fill-line { left center right } 
} 
}
Long text indications can also be automatically wrapped accordingly to the given line width. These will be either left-aligned or justified, as shown in the following example.

\markup {
  \column {
    \line \smallCaps { La vida breve }
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de 
gitanos en el Albaicín de Granada. Al fondo una 
puerta por la que se ve el negro interior de 
una Fragua, iluminado por los rojos resplandores 
del fuego.)
    }
  }
  \hspace #0

  \line \bold { Acto II }
  \override #'(line-width . 50)
  \justify \italic {
    (Calle de Granada. Fachada de la casa de Carmela 
y su hermano Manuel con grandes ventanas abiertas 
a través de las que se ve el patio 
donde se celebra una alegre fiesta)
  }
}
}

LA VIDA BREVE

**Acto I**

*(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)*

**Acto II**

*(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre fiesta)*

An exhaustive list of text alignment commands can be found in Section A.11.2 [Align], page 721.

**See also**

Notation Reference: Section A.11.2 [Align], page 721, [Text marks], page 248.
Installed Files: *scm/define-markup-commands.scm*. 
Graphic notation inside markup

Various graphic objects may be added to a score, using markup commands.

Some markup commands allow decoration of text elements with graphics, as demonstrated in the following example.

```
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
    }
    \null
    \rounded-box \bold Prelude
  }
}
```

![Graphic notation inside markup](image)

Erik Satie [1866 - 1925]

Prelude

Some commands may require an increase in the padding around the text; this is achieved with some markup commands exhaustively described in Section A.11.2 [Align], page 721.

```
\markup \fill-line {
  \center-column {
    \box "Charles Ives (1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
    \bold "Largo to Presto"
  }
  \pad-around #3
    "String quartet keeps very even time,
    Flute quartet keeps very uneven time."
```
Charles Ives (1874 - 1954)

THE UNANSWERED QUESTION
A Cosmic Landscape

Largo to Presto

String quartet keeps very even time, Flute quartet keeps very uneven time.

Other graphic elements or symbols may be printed without requiring any text. As with any markup expression, such objects can be combined.

\markup {
\combine
\draw-circle #4 #0.4 ##f
\filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
\hspace #5
\center-column {
\triangle ##t
\combine
\draw-line #'(0 . 4)
\arrow-head #Y #DOWN ##f
}
}

Advanced graphic features include the ability to include external image files converted to the Encapsulated PostScript format (.eps), or to directly embed graphics into the input file, using native PostScript code. In such a case, it may be useful to explicitly specify the size of the drawing, as demonstrated below:

c'1``\markup {
\combine
\epsfile #X #10 "/context-example.eps"
\with-dimensions #'(0 . 6) #'(0 . 10)
\postscript "
\-2 \3 translate
2.7 2 scale
newpath
2 \-1 moveto
4 \-2 4 1 1 arct
4 2 3 3 1 arct
"
Music notation inside markup

Various musical notation elements may be added to a score, inside a markup object. Notes and accidentals can be entered using markup commands:

```latex
a'2 \markup {
\note {4} #1
\note-by-number #1 #1 #1.5
}
b'1 \markup {
\natural \semiflat \flat
\sesquiflat \doubleflat
}
\glissando
a'1 \markup {
\natural \semisharp \sharp
\sesquisharp \doublesharp
}
\glissando b'
```

An exhaustive list of graphics-specific commands can be found in Section A.11.3 [Graphic], page 736.

See also

Notation Reference: Section A.11.2 [Align], page 721, Section 5.4.4 [Dimensions], page 636, Section 1.7 [Editorial annotations], page 225, Section A.11.3 [Graphic], page 736.

Installed Files: scm/define-markup-commands.scm, scm/stencil.scm.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.
Other notation objects may also be printed in markup mode:

\relative {
  g\prime \bes
  ees\finger \markup \tied-lyric "4\~1"
  fis_\markup { \dynamic rf }
  \tie \markup { \beam #8 #0.1 #0.5 }
  cis
}

c\prime 4

More generally, any available musical symbol may be included separately in a markup object, as demonstrated below; an exhaustive list of these symbols and their names can be found in Section A.8 [The Emmentaler font], page 683.

\relative {
  c\prime 2
  c\prime \markup { \musicglyph "eight" }
  c, 4
  c, 8 \markup { \musicglyph "clefs.G_change" }
  c16
  c2\markup { \musicglyph "timesig.neomensural94" }
}

Another way of printing non-text glyphs is described in [Fonts explained], page 267. This is useful for printing braces of various sizes.

The markup mode also supports diagrams for specific instruments:

\relative {
  c\prime 1\markup {
    \fret-diagram-terse "x;x;o;2;3;2;"
  }
  c\markup { \harp-pedal "\~v|--ov^" }
  c
}


\begin{verbatim}
\combine
\musicglyph "accordion.discant"
\combine
\raise #0.5 \musicglyph "accordion.dot"
\raise #1.5 \musicglyph "accordion.dot"
\end{verbatim}

Such diagrams are documented in Section A.11.5 [Instrument Specific Markup], page 750.

A whole score can even be nested inside a markup object:

\begin{verbatim}
relative {
  c'4 d\markup {
    \score {
      \relative { c'4 d e f }
    }
  }
  e f |
  c d e f
}
\end{verbatim}

An exhaustive list of music notation related commands can be found in Section A.11.4 [Music], page 744.

See also

Notation Reference: Section A.11.4 [Music], page 744, Section A.8 [The Emmentaler font], page 683, [Fonts explained], page 267.

Installed Files: scm/define-markup-commands.scm, scm/fret-diagrams.scm, scm/harp-pedals.scm.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

Known issues and warnings

Vertical spacing of a \score inside a markup object is controlled by baseline-skip. Any \paper settings are ignored.
Multi-page markup

Although standard markup objects are not breakable, a specific syntax makes it possible to enter lines of text that can spread over multiple pages:

\markuplist {
  \justified-lines {
    A very long text of justified lines.
    ...
  }
  \wordwrap-lines {
    Another very long paragraph.
    ...
  }
  ...
}

A very long text of justified lines. ...

Another very long paragraph. ...

This syntax accepts a list of markups, that can be

- the result of a markup list command,
- a list of markups,
- a list of markup lists.

An exhaustive list of markup list commands can be found in Section A.12 [Text markup list commands], page 765.

See also

Notation Reference: Section A.12 [Text markup list commands], page 765.
Extending LilyPond: Section “New markup list command definition” in Extending.
Installed Files: scm/define-markup-commands.scm.
Snippets: Section “Text” in Snippets.
Internals Reference: Section “TextScript” in Internals Reference.

Predefined commands

\markuplist.

1.8.3 Fonts

This section presents the way fonts are handled, and how they may be changed in scores.

Fonts explained

Fonts are handled through several libraries. FontConfig is used to detect available fonts; the selected fonts are rendered using Pango.

In addition to any font already installed on the operating system, additional fonts may be added to the set detected by FontConfig (and thus available in LilyPond scores) through the following commands:

#(ly:font-config-add-font "path/to/font-file.otf")
Both of these commands accept either absolute or relative paths, which makes it possible to compile a score on any system by simply copying the relevant font files alongside the source code.

Having been detected, fonts must then be selected using the methods described in [Single entry fonts], page 271, and [Entire document fonts], page 271. As it prints the complete list of available fonts in the console log, the command `#(ly:font-config-display-fonts)` may be of use to verify which fonts are indeed detected, and under which names (as these may differ from the file names themselves).

Music notation fonts can be described as a set of specific glyphs, ordered in several families. The following syntax allows LilyPond’s various Feta glyphs to be used directly in markup mode:

```ly
\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup "brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup "noteheads.s0petrucci"
  }
}
```

However, all these glyphs except the braces of various sizes contained in `fetaBraces` are available using the simpler syntax described in [Music notation inside markup], page 264.

When using the glyphs contained in `fetaBraces`, the size of the brace is specified by the numerical part of the glyph name, in arbitrary units. Any integer from 0 to 575 inclusive may be specified, 0 giving the smallest brace. The optimum value must be determined by trial and error. These glyphs are all left braces; right braces may be obtained by rotation, see Section 5.4.9 [Rotating objects], page 648.

Three families of text fonts are made available: the `roman` (serif) font, the `sans` font, and the `typewriter` font.

For **svg** backend:

<table>
<thead>
<tr>
<th>Family</th>
<th>Default font</th>
</tr>
</thead>
<tbody>
<tr>
<td>roman</td>
<td>serif</td>
</tr>
<tr>
<td>sans</td>
<td>sans-serif</td>
</tr>
<tr>
<td>typewriter</td>
<td>monospace</td>
</tr>
</tbody>
</table>

`serif`, `sans-serif`, and `monospace` are **generic-family** in SVG and CSS specifications.

For other backends:
LilyPond Serif, LilyPond Sans Serif, and LilyPond Monospace are font aliases defined in the LilyPond dedicated FontConfig configuration file 00-lilypond-fonts.conf. Where a character doesn’t exist in the first font listed, the next font listed will be used instead for that character. For details of alias definitions, please see to 00-lilypond-fonts.conf under the installed directory.

Each family may include different shapes and series. The following example demonstrates the ability to select alternate families, shapes, series and sizes. The value supplied to font-size is the required change from the default size.

```lilypond
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d''2.\markup "Allegro"
\override Voice.TextScript.font-size = #-3
c''4^smaller
```

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{example.png}
\caption{Example output}
\end{figure}

A similar syntax may be used in markup mode; however in this case it is preferable to use the simpler syntax explained in [Selecting font and font size], page 254:

```lilypond
\markup {
\column {
\line {
\override #'{(font-shape . italic) (font-size . 4))
Idomeneo,
}
\line {
\override #'{(font-family . typewriter)
{
\override #'{(font-series . bold)
re
di
}
\override #'{(font-family . sans)
Creta
}
}
}
```

```
When using OpenType fonts, font features can be used. OpenType font scripts and languages cannot be specified. Note: Not all OpenType fonts have all features. If you request a feature that doesn’t exist in the chosen font, then the feature is simply ignored. These examples use TeX Gyre Schola.

% True small caps
\markup { \override #'(font-name . "TeX Gyre Schola") Normal Style: Hello HELLO }
\markup { \override #'(font-name . "TeX Gyre Schola") \caps { Small Caps: Hello } }
\markup { \override #'(font-name . "TeX Gyre Schola") \override #'(font-features . ("smcp")) \caps { True Small Caps: Hello } }

% Number styles
\markup { \override #'(font-name . "TeX Gyre Schola") Normal Number Style: 0123456789 }
\markup { \override #'(font-name . "TeX Gyre Schola") \override #'(font-features . ("onum")) \caps { Old Number Style: 0123456789 } }

% Stylistic Alternates
\markup { \override #'(font-name . "TeX Gyre Schola") \override #'(font-features . ("salt")) \caps { Stylistic Alternates 0: \epsilon\phi\pi\rho\theta } }
\markup { \override #'(font-name . "TeX Gyre Schola") \override #'(font-features . ("salt")) \caps { Stylistic Alternates 1: \epsilon\phi\pi\rho\theta } }

% Multiple features
\markup { \override #'(font-name . "TeX Gyre Schola") \override #'(font-features . ("onum")) \caps { Multiple features: Hello 0123456789 \epsilon\phi\pi\rho\theta } }

Normal Style: Hello HELLO

SMALL CAPS: HELLO

TRUE SMALL CAPS: HELLO

Normal Number Style: 0123456789

Old Number Style: 0123456789

Stylistic Alternates 0: \epsilon\phi\pi\rho\theta

Stylistic Alternates 1: \epsilon\phi\pi\rho\theta

MULTIPLE FEATURES: HELLO 0123456789 \epsilon\phi\pi\rho\theta

For the full OpenType font feature list please see: https://www.microsoft.com/typography/otspec/featurelist.htm

For identifying features of OpenType fonts please see: http://lists.gnu.org/archive/html/lilypond-devel/2017-08/msg00004.html

Although it is easy to switch between preconfigured fonts, it is also possible to use other fonts, as explained in the following sections: [Single entry fonts], page 271, and [Entire document fonts], page 271.
See also

Notation Reference: Section A.8 [The Emmentaler font], page 683, [Music notation inside markup], page 264, Section 5.4.9 [Rotating objects], page 648, [Selecting font and font size], page 254, Section A.11.1 [Font], page 711.

Single entry fonts

Any font that is installed on the operating system and recognized by FontConfig may be used in a score, using the following syntax:

```
\override Staff.TimeSignature.font-name = "Bitstream Charter"
\override Staff.TimeSignature.font-size = #2
\time 3/4
```

```
a'1\markup {
  \override #'(font-name . "Bitstream Vera Sans,sans-serif, Oblique Bold")
  { Vera Oblique Bold }
}
```

```
\relative c'\{
  c1-\markup {
    \override #'(font-name . "Bitstream Vera Sans,sans-serif, Oblique Bold")
    { Vera Oblique Bold }
  }
```

`font-name` can be described using a comma-separated list of ‘fonts’ and a white-space separated list of ‘styles’. As long as the ‘font’ in the list is installed and contains requested glyph, it will be used, otherwise the next font in the list will be used instead.

Running lilypond with the following option displays a list of all available fonts on the operating system:

```
lilypond -dshow-available-fonts
```

See also

Notation Reference: [Fonts explained], page 267, [Entire document fonts], page 271.

Snippets: Section “Text” in Snippets.

Entire document fonts

It is possible to change the fonts to be used as the default fonts in the roman, sans and typewriter font families by specifying them, in that order, as shown in the example below, which automatically scales the fonts with the value set for the global staff size. Similar to [Single entry fonts], page 271, it can be described using a comma-separated list of ‘fonts’. However, font ‘styles’ can not be described. For an explanation of fonts, see [Fonts explained], page 267.

```
\paper { 
  #(define fonts
    (make-pango-font-tree "Times New Roman"
      "Nimbus Sans,Nimbus Sans L"
      "Luxi Mono"
    (/ staff-height pt 20)))
}
```

```
\relative c'\{
  c1-\markup {
    roman,
    \sans sans,
```

```
Note: make-pango-font-tree resets the notation fonts to default of Emmentaler.

The following syntax allows you to change the specific fonts, leaving others to the default values. The following example has the same effect as the above make-pango-font-tree example. The same as make-pango-font-tree, you can specify a comma-separated list of ‘fonts’ for roman, sans, and typewriter categories. If you don’t change the staff size from the default of 20 pt, #:factor (/ staff-height pt 20) is unnecessary.

\paper {
  #(define fonts
    (set-global-fonts
      #:roman "Times New Roman"
      #:sans "Nimbus Sans,Nimbus Sans L"
      #:typewriter "Luxi Mono"
      #:factor (/ staff-height pt 20) ; unnecessary if the staff size is default
    ))
}

It also can specify the notation fonts. The following example also has the same effect as the above examples, because it sets the notation fonts as the default. For more information, see Section 3.4.4 [Replacing the notation font], page 529.

\paper {
  #(define fonts
    (set-global-fonts
      #:music "emmentaler" ; default
      #:brace "emmentaler" ; default
      #:roman "Times New Roman"
      #:sans "Nimbus Sans,Nimbus Sans L"
      #:typewriter "Luxi Mono"
      #:factor (/ staff-height pt 20) ; unnecessary if the staff size is default
    ))
}

Note: Each call to set-global-fonts completely resets both the main notation and text fonts. If any category is left unspecified, then the default font will be used for that category. Each call of set-global-fonts affects each \book blocks that follow it. If there are multiple \book blocks and you want to use different fonts for each, simply call set-global-fonts again, like this:

\paper {
  #(define fonts
    (set-global-fonts
      ...
    ))
}

\book {
  ...
}
\paper {
  #(define fonts
    (set-global-fonts
      ...
    ))
}
\book {
  ...
}

**See also**

Notation Reference: [Fonts explained], page 267, [Single entry fonts], page 271, [Selecting font and font size], page 254, Section A.11.1 [Font], page 711, Section 3.4.4 [Replacing the notation font], page 529.
2 Specialist notation

This chapter explains how to create musical notation for specific types of instrument or in specific styles.

Orchestral and ensemble music is not addressed in a separate section; however, several notions commonly useful in orchestral scores are found in [References for opera and stage musicals], page 318.

2.1 Vocal music

This section explains how to typeset vocal music, and make sure that the lyrics will be aligned with the notes of their melody.

2.1.1 Common notation for vocal music

This section discusses issues common to most types of vocal music.

References for vocal music

This section indicates where to find details of notation issues that may arise in any type of vocal music.

- Most styles of vocal music use written text as lyrics. An introduction to this notation is to be found in Section “Setting simple songs” in Learning Manual.
- Vocal music is likely to require the use of markup mode, either for lyrics or for other text elements (characters’ names, etc.). This syntax is described in [Text markup introduction], page 252.
- Ambitus may be added at the beginning of vocal staves, as explained in [Ambitus], page 36.
- Dynamic markings by default are placed below the staff, but in choral music they are usually placed above the staff in order to avoid the lyrics, as explained in [Score layouts for choral], page 315.
See also

- Notation Reference: [Text markup introduction], page 252, [Ambitus], page 36, [Score layouts for choral], page 315.
- Snippets: Section “Vocal music” in Snippets.

**Entering lyrics**

Lyrics are entered in a special input mode, which can be introduced by the keyword `\lyricmode`, or by using `\addlyrics` or `\lyricsto`. In this special input mode, the input `d` is not parsed as the pitch $D$, but rather as a one-letter syllable of text. In other words, syllables are entered like notes but with pitches replaced by text.

For example:

```latex
\lyricmode { Three blind mice, two three blind mice }
```

There are two main methods for specifying the horizontal placement of the syllables, either by specifying the duration of each syllable explicitly, as in the example above, or by leaving the lyrics to be aligned automatically to a melody or other voice of music, using `\addlyrics` or `\lyricsto`. The former method is described below in [Manual syllable durations], page 280. The latter method is described in [Automatic syllable durations], page 278.

A word or syllable of lyrics begins with an alphabetic character (plus some other characters, see below) and is terminated by any white space or a digit. Later characters in the syllable can be any character that is not a digit or white space.

Because any character that is not a digit or white space is regarded as part of the syllable, a word is valid even if it ends with `,`, which often leads to the following mistake:

```latex
\lyricmode { lah lah lah }
```

In this example, the `}` is included in the final syllable, so the opening brace is not balanced and the input file will probably not compile. Instead, braces should always be surrounded with white space:

```latex
\lyricmode { lah lah lah }
```

Punctuation, lyrics with accented characters, characters from non-English languages, or special characters (such as the heart symbol or slanted quotes), may simply be inserted directly into the input file, providing it is saved with UTF-8 encoding. For more information, see Section 3.3.3 [Special characters], page 524.

```latex
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { ""I" am so lonely, "" said she }
```

Normal quotes may be used in lyrics, but they have to be preceded with a backslash character and the whole syllable has to be enclosed between additional quotes. For example,

```latex
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { ""I" am so lonely, "" said she }
```
The full definition of a word start in lyrics mode is somewhat more complex. A word in lyrics mode is one that begins with an alphabetic character, _, ?, !, ;, ’, the control characters ^A through ^F, ^Q through ^W, ^Y, ^^, any 8-bit character with an ASCII code over 127, or a two-character combination of a backslash followed by one of \, ’, ”, or ^.

Great control over the appearance of lyrics comes from using \markup inside the lyrics themselves. For explanation of many options, see Section 1.8.2 [Formatting text], page 252.

Selected Snippets

**Formatting lyrics syllables**

Markup mode may be used to format individual syllables in lyrics.

```latex
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}
```

```
<<
  \new Voice = melody \mel
  \new Lyrics \lyricstomelody \lyr
>>
```

Lyrics can contain Markup!

See also


Notation Reference: [Automatic syllable durations], page 278, Section 1.8.3 [Fonts], page 267, Section 1.8.2 [Formatting text], page 252, Section 5.4.1 [Input modes], page 632, [Manual syllable durations], page 280, Section 3.3.3 [Special characters], page 524.

Internals Reference: Section “LyricText” in Internals Reference.

Snippets: Section “Text” in Snippets.

Aligning lyrics to a melody

Lyrics are interpreted in \lyricmode and printed in a Lyrics context, see Section 5.1.1 [Contexts explained], page 597.

```latex
\new Lyrics \lyricstomelody { ... }
```

Two variants of \lyricmode additionally set an associated context used to synchronise the lyric syllables to music. The more convenient \addlyrics immediately follows the musical content of the Voice context with which it should be synchronised, implicitly creating a Lyrics context of its own. The more versatile \lyricsto requires both specifying the associated Voice context by name and explicitly creating a containing Lyrics context. For details see [Automatic syllable durations], page 278.

Lyrics can be aligned with melodies in two main ways:

- Lyrics can be aligned automatically, with the durations of the syllables being taken from another voice of music or (in special circumstances) an associated melody, using \addlyrics,
\lyricsto, or by setting the \associatedVoice property. For more details, see [Automatic syllable durations], page 278.

\begin{verbatim}
\new Staff <<
\time 2/4
\new Voice = "one" \relative {
  \voiceOne
  c''4 b8. a16 g4. r8 a4 ( b ) c2
}
\new Voice = "two" \relative {
  \voiceTwo
  s2 s4. f'8 e4 d c2
}
>>

% takes durations and alignment from notes in "one"
\new Lyrics \lyricsto "one" {
  Life is __ _ love, live __ life.
}

% takes durations and alignment from notes in "one" initially
% then switches to "two"
\new Lyrics \lyricsto "one" {
  No more let
  \set associatedVoice = "two" % must be set one syllable early
  sins and sor -- rows grow.
}
\end{verbatim}

Life is__ love, live__ life.
No more let sins and sor-- rows grow.

The first stanza shows the normal way of entering lyrics.
The second stanza shows how the voice from which the lyric durations are taken can be changed. This is useful if the words to different stanzas fit the notes in different ways and all the durations are available in Voice contexts. For more details, see Section 2.1.3 [Stanzas], page 306.

• Lyrics can be aligned independently of the duration of any notes if the durations of the syllables are specified explicitly, and entered with \lyricmode.

\begin{verbatim}
\new Voice = "one" \relative {
  \time 2/4
  c''4 b8. a16 g4. f8 e4 d c2
}

% uses previous explicit duration of 2;
\new Lyrics \lyricmode {
\end{verbatim}
Joy to the earth!
}

% explicit durations, set to a different rhythm
\new Lyrics \lyricmode {
   Life4 is love,2. live4 life.2
}

Joy to the earth!
Life is love, live life.

The first stanza is not aligned with the notes because the durations were not specified, and the previous value of 2 is used for each word.

The second stanza shows how the words can be aligned quite independently from the notes. This is useful if the words to different stanzas fit the notes in different ways and the required durations are not available in a music context. For more details see [Manual syllable durations], page 280. This technique is also useful when setting dialogue over music; for examples showing this, see [Dialogue over music], page 324.

See also


Notation Reference: Section 5.1.1 [Contexts explained], page 597, [Automatic syllable durations], page 278, Section 2.1.3 [Stanzas], page 306, [Manual syllable durations], page 280, [Dialogue over music], page 324, [Manual syllable durations], page 280.

Internals Reference: Section “Lyrics” in Internals Reference.

Automatic syllable durations

Lyrics can be automatically aligned to the notes of a melody in three ways:

- by specifying the named Voice context containing the melody with \lyricsto,
- by introducing the lyrics with \addlyrics and placing them immediately after the Voice context containing the melody,
- by setting the associatedVoice property, the alignment of the lyrics may be switched to a different named Voice context at any musical moment.

In all three methods hyphens can be drawn between the syllables of a word and extender lines can be drawn beyond the end of a word. For details, see [Extenders and hyphens], page 286.

The Voice context containing the melody to which the lyrics are being aligned must not have “died”, or the lyrics after that point will be lost. This can happen if there are periods when that voice has nothing to do. For methods of keeping contexts alive, see Section 5.1.3 [Keeping contexts alive], page 602.

Using \lyricsto

Lyrics can be aligned under a melody automatically by specifying the named Voice context containing the melody with \lyricsto:

<<
   \new Voice = "melody" \relative {
a'1 a4. a8 a2
}
\new Lyrics \lyricsto "melody" {
  These are the words
}
>>

These are the words

This aligns the lyrics to the notes of the named Voice context, which must already exist. Therefore normally the Voice context is specified first, followed by the Lyrics context. The lyrics themselves follow the \lyricsto command. The \lyricsto command invokes lyric mode automatically. By default, the lyrics are placed underneath the notes. For other placements, see [Placing lyrics vertically], page 289.

Using \addlyrics

The \addlyrics command is just a convenient shortcut that can sometimes be used instead of having to set up the lyrics through a more complicated LilyPond structure.

{ MUSIC }
\addlyrics { LYRICS }

is the same as

\new Voice = "blah" { MUSIC }
\new Lyrics \lyricsto "blah" { LYRICS }

Here is an example,

{ 
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}

play the game

More stanzas can be added by adding more \addlyrics sections:

{ 
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
}
The command \addlyrics cannot handle polyphonic settings. Also, it cannot be used to associate lyrics to a \TabVoice. For these cases one should use \lyricsto.

**Using associatedVoice**

The melody to which the lyrics are being aligned can be changed by setting the associatedVoice property,

\[
\set associatedVoice = "lala"
\]

The value of the property (here: "lala") should be the name of a Voice context. For technical reasons, the \set command must be placed one syllable before the one to which the change in voice is to apply.

Here is an example demonstrating its use:

\[
\begin{verbatim}
<<
\new Staff <<
\time 2/4
\new Voice = "one" \relative { \voiceOne
c'1'4 b8. a16 g4. r8 a4 ( b ) c2}
\new Voice = "two" \relative { \voiceTwo
 s2 s4. f'8 e8 d4. c2}
>>
\new Lyrics \lyricsto "one" {
No more let
\set associatedVoice = "two" % must be set one syllable early
sins and sor -- rows grow.
}
>>
\end{verbatim}
\]

See also

Notation Reference: [Extenders and hyphens], page 286, Section 5.1.3 [Keeping contexts alive], page 602, [Placing lyrics vertically], page 289.

**Manual syllable durations**

In some complex vocal music, it may be desirable to place lyrics completely independently of notes. In this case do not use \lyricsto or \addlyrics and do not set associatedVoice.
Syllables are entered like notes – but with pitches replaced by text – and the duration of each syllable is entered explicitly after the syllable.

Hyphenated lines may be drawn between syllables as usual, but extender lines cannot be drawn when there is no associated voice.

Here are two examples:

\begin{verbatim}
\new Voice = "melody" \relative {
    c''2 a f f e e
}\n\new Lyrics \lyricmode {
    c4. -- a -- f -- f -- e2. -- e
}\n\end{verbatim}

This technique is useful when writing dialogue over music, see [Dialogue over music], page 324.

To change syllable alignment, simply override the \texttt{self-alignment-X} property:

\begin{verbatim}
\new Voice = "melody" \relative {
    \time 3/4
\end{verbatim}
Multiple syllables to one note

In order to assign more than one syllable to a single note with spaces between the syllables, you can surround the phrase with quotes or use a _ character. Alternatively, you can use the tilde symbol (~) to get a lyric tie.

\addlyrics
{ \override LyricHyphen.minimum-distance = #1.0 % Ensure hyphens are visible 
  Che in ques -- ta e in quel -- l'al -- tr'on -- da 
}
\addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
\addlyrics { Che e in ques -- ta e in quel -- l'al -- tr'on -- da }

See also

When a melisma occurs on a syllable other than the last one in a word, that syllable is usually joined to the following one with a hyphenated line. This is indicated by placing a double hyphen, --, immediately after the syllable.

Alternatively, when a melisma occurs on the last or only syllable in a word an extender line is usually drawn from the end of the syllable to the last note of the melisma. This is indicated by placing a double underscore, __, immediately after the word.

There are five ways in which melismata can be indicated:

- Melismata are created automatically over notes which are tied together:

  ```plaintext
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g2 ~ |
    4 e2 ~ |
    8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
  ```

- Melismata can be created automatically from the music by placing slurs over the notes of each melisma. This is the usual way of entering lyrics:

  ```plaintext
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
  ```

  Note that phrasing slurs do not affect the creation of melismata.

- Notes are considered a melisma if they are manually beamed, providing automatic beaming is switched off. See [Setting automatic beam behavior], page 87.

  ```plaintext
  \new Voice = "melody" \relative {
    \time 3/4
    \autoBeamOff
    f''4 g8[ f e f]
  ```
Clearly this is not suited to melismata over notes which are longer than eighth notes.

- An unslurred group of notes will be treated as a melisma if they are bracketed between \melisma and \melismaEnd.

- A melisma can be defined entirely in the lyrics by entering a single underscore character, _, for every extra note that has to be added to the melisma.
It is possible to have ties, slurs and manual beams in the melody without their indicating melismata. To do this, set `melismaBusyProperties`:

<<
\new Voice = "melody" \relative {
  \time 3/4
  \set melismaBusyProperties = #'()
  c'4 d ( e )
  g8 [ f ] f4 ~ 4
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e e -- le -- i -- son
}
>>

Other settings for `melismaBusyProperties` can be used to selectively include or exclude ties, slurs, and beams from the automatic detection of melismata; see `melismaBusyProperties` in Section “Tunable context properties” in Internals Reference.

Alternatively, if all melismata indications are to be ignored, `ignoreMelismata` may be set true; see [Stanzas with different rhythms], page 308.

If a melisma is required during a passage in which `melismaBusyProperties` is active, it may be indicated by placing a single underscore in the lyrics for each note which should be included in the melisma:

<<
\new Voice = "melody" \relative {
  \time 3/4
  \set melismaBusyProperties = #'()
  c'4 d ( e )
  g8 [ f ] ~ 4 ~ f
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ e _ _ _ _
}
>>

Predefined commands
\autoBeamOff, \autoBeamOn, \melisma, \melismaEnd.
See also

Notation Reference: [Aligning lyrics to a melody], page 276, [Automatic syllable durations], page 278, [Setting automatic beam behavior], page 87, [Stanzas with different rhythms], page 308.
Internals Reference: Section “Tunable context properties” in Internals Reference.

Known issues and warnings

Extender lines under melismata are not created automatically; they must be inserted manually with a double underscore.

Extenders and hyphens

In the last syllable of a word, melismata are sometimes indicated with a long horizontal line starting in the melisma syllable, and ending in the next one. Such a line is called an extender line, and it is entered as ‘__’ (note the spaces before and after the two underscore characters).

Note: Melismata are indicated in the score with extender lines, which are entered as one double underscore; but short melismata can also be entered by skipping individual notes, which are entered as single underscore characters; these do not make an extender line to be typeset by default.

Centered hyphens are entered as ‘--’ between syllables of a same word (note the spaces before and after the two hyphen characters). The hyphen will be centered between the syllables, and its length will be adjusted depending on the space between the syllables.

In tightly engraved music, hyphens can be removed. Whether this happens can be controlled with the minimum-distance (minimum distance between two syllables) and the minimum-length (threshold below which hyphens are removed) properties of LyricHyphen.

By default a hyphen is not repeated after a system break when the next line begins with a new syllable. Setting the after-line-breaking property to #t allows hyphens to be drawn in such situations.

See also

Internals Reference: Section “LyricExtender” in Internals Reference, Section “LyricHyphen” in Internals Reference.

Gradual changes of vowel

Vowel transitions (gradual changes of vowel or sustained consonant), which may be indicated by arrows between syllables, are entered with the command \vowelTransition (see Gould pp. 452–453). The arrow shows the length of the transition, and it is by default always drawn (space is added if necessary in tightly engraved music). Ties between notes of unchanged pitch or slurs show that there is no new articulation, despite a change of vowel. The minimum length of the arrows may be adjusted with the minimum-length property of VowelTransition.

```latex
\{ 
c'2 ~ c'
\set melismaBusyProperties = #'()
c'2 ~ c'
\time 3/2
```
d'4( e' f' e' f'2
  e'2.)
}\addlyrics
{
  Ah \vowelTransition oh,
  ah \vowelTransition oh,
  ah \vowelTransition _ _ _
  oh.
}

See also
Internals Reference: Section “VowelTransition” in Internals Reference.

2.1.2 Techniques specific to lyrics

Working with lyrics and variables
Variables containing lyrics can be created, but the lyrics must be entered in lyric mode:

\begin{verbatim}
musicOne = \relative {
  c''4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  \new Voice = "one" {
    \time 2/4
    \musicOne
  }
  \new Lyrics \lyricsto "one" {
    \verseOne
  }
\}
\end{verbatim}

Durations do not need to be added if the variable is to be invoked with \addlyrics or \lyricsto.
For different or more complex orderings, the best way is to define the music and lyric variables
first, then set up the hierarchy of staves and lyrics, omitting the lyrics themselves, and then add
the lyrics using \context underneath. This ensures that the voices referenced by \lyricsto have always been defined earlier. For example:

```latex
sopranoMusic = \relative { c'4 c c c }  
contraltoMusic = \relative { a'4 a a a }  
sopranoWords = \lyricmode { Sop -- ra -- no words }  
contraltoWords = \lyricmode { Con -- tral -- to words }  
```

\score {
\new ChoirStaff <<
  \new Staff {
    \new Voice = "sopranos" {
      \sopranoMusic
    }  
  }  
  \new Lyrics = "sopranos"
\new Staff {
  \new Voice = "contraltos" {
    \contraltoMusic
  }  
  \new Lyrics = "contraltos"
  \context Lyrics = "sopranos" {
    \lyricsto "sopranos" {
      \sopranoWords
    }
  }  
  \context Lyrics = "contraltos" {
    \lyricsto "contraltos" {
      \contraltoWords
    }
  }
} >>
}
```

![Musical notation example]

**See also**

Notation Reference: [Placing lyrics vertically], page 289.

Placing lyrics vertically

Depending on the type of music, lyrics may be positioned above the staff, below the staff, or between staves. Placing lyrics below the associated staff is the easiest, and can be achieved by simply defining the Lyrics context below the Staff context:

\score { \newStaff {} \newVoice = "melody" { \relative { c''4 c c c } } \newLyrics { \lyricsto "melody" { Here are the words } } } \newStaff = "staff" { \newVoice = "melody" { \relative { c'4 c c c } } \newLyrics \with { alignAboveContext = "staff" } { \lyricsto "melody" { Here are the words } } } Alternatively, a two-step process may be used. First the Lyrics context is declared (without any content) before the Staff and Voice contexts, then the \lyricsto command is placed after the Voice declaration it references by using \context, as follows:

\score { \newStaff {} \newVoice = "melody" { \relative { c''4 c c c } } \newLyrics \with { alignAboveContext = "staff" } { \lyricsto "melody" { Here are the words } } }
When there are two voices on separate staves the lyrics may be placed between the staves using either of these methods. Here is an example of the second method:

\score {
\new ChoirStaff <<
\new Staff {
  \new Voice = "sopranos" {
    \relative { c'4 c c c }
  }
}
\new Lyrics = "sopranos"
\new Lyrics = "contraltos" \with {
  % lyrics above a staff should have this override
  \override VerticalAxisGroup.staff-affinity = #DOWN
}
\new Staff {
  \new Voice = "contraltos" {
    \relative { a'4 a a a }
  }
}
\context Lyrics = "sopranos" {
  \lyricsto "sopranos" {
    Sop -- ra -- no words
  }
}
\context Lyrics = "contraltos" {
  \lyricsto "contraltos" {
    Con -- tral -- to words
  }
}
>>
}
Other combinations of lyrics and staves may be generated by elaborating these examples, or by examining the templates in the Learning Manual, see Section “Vocal ensembles templates” in Learning Manual.

Selected Snippets

*Obtaining 2.12 lyrics spacing in newer versions*

The vertical spacing engine changed since version 2.14. This can cause lyrics to be spaced differently.

It is possible to set properties for Lyric and Staff contexts to get the spacing engine to behave as it did in version 2.12.

```latex
global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

words = \lyricmode {
  Great is Thy faith -- ful -- ness,
}

\score {
  \new ChoirStaff <<
    \new Staff = sopranos
    \new Lyrics = sopranos
    \new Staff = women <<
```
\new Voice = "sopranos" {
    \voiceOne
    \global \sopMusic
}
\new Voice = "altos" {
    \voiceTwo
    \global \altoMusic
}

\new Lyrics = "altos"
\new Lyrics = "tenors"
\new Staff = men <<
    \clef bass
\new Voice = "tenors" {
    \voiceOne
    \global \tenorMusic
}
\new Voice = "basses" {
    \voiceTwo \global \bassMusic
}

\new Lyrics = basses
\context Lyrics = sopranos \lyricsto sopranos \words
\context Lyrics = altos \lyricsto altos \words
\context Lyrics = tenors \lyricsto tenors \words
\context Lyrics = basses \lyricsto basses \words

\layout {
    \context {
        \Lyrics
        \override VerticalAxisGroup.staff-affinity = ##f
        \override VerticalAxisGroup.staff-staff-spacing = #'((basic-distance . 0)
            (minimum-distance . 2)
            (padding . 2))
    }
    \context {
        \Staff
        \override VerticalAxisGroup.staff-staff-spacing = #'((basic-distance . 0)
            (minimum-distance . 2)
            (padding . 2))
    }
}
See also


Notation Reference: Section 5.1.7 [Context layout order], page 614, Section 5.1.2 [Creating and referencing contexts], page 599.

Placing syllables horizontally

To increase the spacing between lyrics, set the minimum-distance property of LyricSpace.

```
\relative c' {
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```
To make this change for all lyrics in the score, set the property in the \layout block.

\score {
  \relative { c' c c c c c c c }
  \addlyrics {
    longtext longtext longtext longtext longtext longtext longtext longtext
  }
  \layout {
    \context { \Lyrics
      \override LyricSpace.minimum-distance = #1.0
    }
  }
}

Selected Snippets

Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the self-alignment-X property of the LyricText object. #1 is left, #0 is center and #1 is right; however, you can use #LEFT, #CENTER and #RIGHT as well.

\layout { ragged-right = ##f }
\relative c'' {
  c1 c1 c1 c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}
Known issues and warnings

Checking to make sure that text scripts and lyrics are within the margins requires additional calculations. To speed up processing slightly, this feature can be disabled:

```
\override Score.PaperColumn.keep-inside-line = ##f
```

To make lyrics avoid bar lines as well, use

```
\layout {
  \context {\Lyrics
    \consists "Bar_engraver"
    \consists "Separating_line_group_engraver"
    \hide BarLine
    }
}
```

Lyrics and repeats

Simple repeats

Repeats in music are fully described elsewhere; see Section 1.4 [Repeats], page 151. This section explains how to add lyrics to repeated sections of music.

Lyrics to a section of music that is repeated should be surrounded by exactly the same repeat construct as the music, if the words are unchanged.

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {a'4 a a a}
        \repeat volta 2 { b4 b b b }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re--peat--ed.
        \repeat volta 2 { Re--peat--ed twice. }
      }
    }
  >>
}
```

The words will then be correctly expanded if the repeats are unfolded.

```
\score {
  \unfoldRepeats {
```

Not repeated. Repeated twice.

This is left-aligned This is centered This is right-aligned
\new Staff {
\new Voice = "melody" {
\relative {
 a'4 a a a
 \repeat volta 2 { b4 b b b }
}
}
}
\new Lyrics {
\lyricsto "melody" {
 Not re -- peat -- ed.
 \repeat volta 2 { Re -- peat -- ed twice. }
}
}
}>


If the repeated section is to be unfolded and has different words, simply enter all the words:
\score {
<<
\new Staff {
\new Voice = "melody" {
\relative {
 a'4 a a a
 \repeat unfold 2 { b4 b b b }
}
}
}
\new Lyrics {
\lyricsto "melody" {
 Not re -- peat -- ed.
 \repeat volta 2 { Re -- peat -- ed twice. }
}
}
}>

Not repeated. The first time words. Second time words.

When the words to a repeated volta section are different, the words to each repeat must be entered in separate \Lyrics contexts, correctly nested in parallel sections:
\score {
More verses may be added in a similar way:
\score { <<
  \new Staff {
    \new Voice = "singleVoice" {
      \relative {
        a'4 a a a
      }
    }
  }
  \new Lyrics \lyricsto "singleVoice" {
    Not re--peat--ed.
    <<
      \set associatedVoice = "singleVoice"
      \measure {2/8}{
        \repeat volta 3 { b4 b b b }
        c4 c c c
      }
    }
    \new Lyrics {
      \set associatedVoice = "singleVoice"
      The third time words.
    }
    \new Lyrics {
      \set associatedVoice = "singleVoice"
      Sec--ond time words.
    }
  }
}
However, if this construct is embedded within a multi-staff context such as a ChoirStaff the lyrics of the second and third verses will appear beneath the bottom staff.

To position them correctly use alignBelowContext:

\score { 
  \new Staff { 
    \new Voice = "melody" { 
      \relative { 
        a'4 a a a 
        \repeat volta 3 { b4 b b b } 
        c4 c c c 
      } 
    } 
  } 
  \new Lyrics = "firstVerse" \lyricsto "melody" { 
    \NotRepeateD. 
    \TheFirstTimeWords. \TheEndSection. 
    \SecondTimeWords. 
    \TheThirdTimeWords. 
  } 
  \new Lyrics = "secondVerse" 
  \with { alignBelowContext = "firstVerse" } { 
    \set associatedVoice = "melody" 
    \SecondTimeWords. 
  } 
  \new Lyrics = "thirdVerse" 
  \with { alignBelowContext = "secondVerse" } { 
    \set associatedVoice = "melody" 
    \TheThirdTimeWords. 
  } 
} 
}
Repeats with alternative endings

If the words of the repeated section are the same, and none of the \alternative blocks start with a rest, exactly the same structure can be used for both the lyrics and music. This has the advantage that \unfoldRepeats will expand both music and lyrics correctly.

\score { << 
\new Staff { 
\time 2/4 
\new Voice = "melody" { 
\relative { 
a'4 a a a 
\repeat volta 2 { b4 b } 
\alternative { { b b } { b c } } } 
} } 
\new Lyrics { 
\lyricsto "melody" { 
Not re -- peat -- ed. 
\repeat volta 2 { Re -- peat -- } 
\alternative { { ed twice. } { ed twice. } } } 
} } >> 
}

But when the repeated section has different words, or when one of the \alternative blocks starts with a rest, a repeat construct cannot be used around the words and \skip commands have to be inserted manually to skip over the notes in the alternative sections which do not apply.

Note: do not use an underscore, \_, to skip notes – an underscore indicates a melisma, causing the preceding syllable to be left-aligned.

\Note: The \skip command must be followed by a number, but this number is ignored in lyrics which derive their durations from the notes in an associated melody through \addlyrics or \lyricsto. Each \skip skips a single note of any value, irrespective of the value of the following number.
When a note is tied over into two or more alternative endings a tie is used to carry the note into the first alternative ending and a \texttt{\repeatTie} is used in the second and subsequent endings. This structure causes difficult alignment problems when lyrics are involved and increasing the length of the alternative sections so the tied notes are contained wholly within them may give a more acceptable result.

The tie creates a melisma into the first alternative, but not into the second and subsequent alternatives, so to align the lyrics correctly it is necessary to disable the automatic creation of melismata over the volta section and insert manual skips.

\score {
\new Staff {
\time 2/4
\new Voice = "melody" {
\relative {
\set melismaBusyProperties = #'() 
\repeat volta 2 { b'4 b ~}
\}}
\new Lyrics {
\lyricsto "melody" {
The first time words.
\repeat unfold 2 { \skip 1 }
End here.
}
\}
\new Lyrics {
\lyricsto "melody" {
Sec -- ond
\repeat unfold 2 { \skip 1 }
time words.
}
}
}\}
}}
Note that if \unfoldRepeats is used around a section containing \repeatTie, the \repeatTie should be removed to avoid both types of tie being printed.

When the repeated section has different words a \repeat cannot be used around the lyrics and \skip commands need to be inserted manually, as before.
Chapter 2: Specialist notation

If you wish to show extenders and hyphens into and out of alternative sections these must be inserted manually.

\score {
<<
\new Staff {
  \time 2/4
  \new Voice = "melody" {
    \relative {
      \repeat volta 2 { b'4 b ~}
      \alternative { { b b } { b \repeatTie c } }
      c4 c
    }
  }
}
\new Lyrics {
  \lyricsto "melody" {
    \repeat unfold 2 { \skip 1 }
  }
}\new Lyrics {
  \lyricsto "melody" {
    Here's "a_
    \skip 1
    "_" sec -- ond one.
  }
}>

Here's a__verse.
Here's one more to sing.

See also
Notation Reference: Section 5.1.3 [Keeping contexts alive], page 602, Section 1.4 [Repeats], page 151.
Divisi lyrics

When just the words and rhythms of the two parts differ with the pitches remaining the same, temporarily turning off the automatic detection of melismata and indicating the melisma in the lyrics may be the appropriate method to use:

\score {
  \new Voice = "melody" {
    \relative c' {
      \set melismaBusyProperties = #'( )
      \slurDown
      \slurDashed
      e4 e8 ( e ) c4 c |
      \unset melismaBusyProperties
      c
    }
  }
  \new Lyrics \lyricsto "melody" {
    They shall not o--ver--come
  }
  \new Lyrics \lyricsto "melody" {
    We will _
  }
}

They shall not over-come
We will

When both music and words differ it may be better to display the differing music and lyrics by naming voice contexts and attaching lyrics to those specific contexts:

\score {
  \new Voice = "melody" {
    \relative {
      <<
      \voiceOne
      e'4 e8 e
    }
    \new Voice = "splitpart" {
      \voiceTwo
      c4 c
    }
    >>
    \oneVoice
    c4 c |
    c
  }
}
It is common in choral music to have a voice part split for several measures. The << {...} \ {...} >> construct, where the two (or more) musical expressions are separated by double backslashes, might seem the proper way to set the split voices. This construct, however, will assign all the expressions within it to NEW Voice contexts which will result in no lyrics being set for them since the lyrics will be set to the original voice context – not, typically, what one wants. The temporary polyphonic passage is the proper construct to use, see section Temporary polyphonic passages in [Single-staff polyphony], page 173.

Polyphony with shared lyrics

When two voices with different rhythms share the same lyrics, aligning the lyrics to one of the voices may lead to problems in the other voice. For example, the second lyric extender below is too short, since the lyrics are aligned only to the top voice:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
words = \lyricmode { la __ la __ }
```

```
\new Staff <<
\new Voice = "sopranoVoice" { \voiceOne \soprano }
\new Voice { \voiceTwo \alto }
\new Lyrics \lyricsto "sopranoVoice" \words
>>
```

To get the desired result, align the lyrics to a new NullVoice context containing a suitable combination of the two voices. The notes of the NullVoice context do not appear on the printed page, but can be used to align the lyrics appropriately:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }
```
This method also can be used with the \partCombine function, which does not allow lyrics on its own:

\begin{verbatim}
\new Staff <<
\new Voice \partCombine \soprano \alto
\new NullVoice = "aligner" \aligner
\new Lyrics \lyricsto "aligner" \words
>>
\end{verbatim}

Known issues and warnings

The \addlyrics function only works with \Voice lyrics and so cannot be used with \NullVoice. The \partCombine function is described in [ Automatic part combining ], page 183.

Lastly, this method can be used even when the voices are in different staves, and is not limited to only two voices:

\begin{verbatim}
\new ChoirStaff \with {\accepts NullVoice } <<
\new Staff \soprano
\new NullVoice = "aligner" \aligner
\new Lyrics \lyricsto "aligner" \words
\new Staff \partCombine \altoOne \altoTwo
>>
\end{verbatim}
2.1.3 Stanzas

Adding stanza numbers

Stanza numbers can be added by setting `stanza`, e.g.,

```latex
\new Voice \relative { 
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics { 
  \set stanza = "1. " 
  Hi, my name is Bert. 
} \addlyrics { 
  \set stanza = "2. " 
  Oh, ché -- ri, je t'aime 
}
```

These numbers are put just before the start of the first syllable. Two lines of a stanza can also be grouped together, for example in case of a repeat with different lyrics:

```latex
leftbrace = \markup { 
  \override #'(font-encoding . fetaBraces) 
  \lookup "brace80"
}

stanzaOneOne = \lyricmode { 
  \set stanza = \markup { 
    \column { \vspace #.33 "1. "} 
    \leftbrace 
  } 
  Child, you're mine and I love you. 
  Lend thine ear to what I say. 
}

stanzaOneThree = \lyricmode { 
  Child, I have no great -- er joy 
  Than to have you walk in truth. 
}

\new Voice { 
\repeat volta 2 { 
  c'8 c' c' c' c' c'4 
  c'8 c' c' c' c' c'4
}
```
Adding dynamics marks to stanzas

Stanzas differing in loudness may be indicated by putting a dynamics mark before each stanza. In LilyPond, everything coming in front of a stanza goes into the StanzaNumber object; dynamics marks are no different. For technical reasons, you have to set the stanza outside \lyricmode:

```lilypond
\addlyrics { \stanzaOneOne }
\addlyrics { \stanzaOneThree }

Child, you're mine and I love you.
Child, I have no greater joy

Lend thine ear to what I say.
Than to have you walk in truth.
```

Adding singers' names to stanzas

Names of singers can also be added. They are printed at the start of the line, just like instrument names. They are created by setting vocalName. A short version may be entered as shortVocalName.

```lilypond
\new Voice \relative {
    \time 3/4
g'4 c'2
} \addlyrics {

ff 1. Big bang
```
\set vocalName = "Bert "
Hi, my name is Bert.
\addlyrics {
\set vocalName = "Ernie ",
Oh, ché -- ri, je t'aime
}

Stanzas with different rhythms
Often, different stanzas of one song are put to one melody in slightly differing ways. Such variations can still be captured with \lyricsto.

Ignoring melismata
One possibility is that the text has a melisma in one stanza, but multiple syllables in another. One solution is to make the faster voice ignore the melisma. This is done by setting ignoreMelismata in the Lyrics context.

\begin{verbatim}
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
c'4
  \slurDotted
  f8.[( g16])
a4
}
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
  \unset ignoreMelismata
  still
  }
>>

Known issues and warnings
Unlike most \set commands, \set ignoreMelismata does not work if prefixed with \once. It is necessary to use \set and \unset to bracket the lyrics where melismata are to be ignored.
Adding syllables to grace notes

By default, grace notes (e.g., via \texttt{\textbackslash grace}) do not get assigned syllables when using \texttt{\textbackslash lyricsto}, but this behavior can be changed:

\begin{verbatim}
<< \new Voice = melody \relative { 
f'4 \appoggiatura a32 b4 
\grace { f16 a16 } b2 
\afterGrace b2 { f16[ a16] } 
\appoggiatura a32 b4 
\acciaccatura a8 b4 
} 
\new Lyrics \lyricsto melody { 
  \set includeGraceNotes = ##t 
  \case,
  gra -- ce case,
  after -- grace case,
  \set ignoreMelismata = ##t 
  app. case,
  acc. case.
} >>
\end{verbatim}

![Musical notation example](image)

Known issues and warnings

Like \texttt{\textbackslash associatedVoice}, \texttt{\textbackslash includeGraceNotes} needs to be set at latest one syllable before the one which is to be put under a grace note. In the case of a grace note at the very beginning of a piece of music, using a \texttt{\textbackslash with}, or a \texttt{\textbackslash context} block within \texttt{\textbackslash layout}, is recommended:

\begin{verbatim}
<< \new Voice = melody \relative c' { 
  \grace { c16( d e f } 
  gi) f 
} 
\new Lyrics \with { includeGraceNotes = ##t } 
\lyricsto melody { 
  Ah __ fa 
} >>
\end{verbatim}

![Musical notation example](image)
Switching to an alternative melody

More complex variations in setting lyrics to music are possible. The melody to which the lyrics are being set can be changed from within the lyrics by setting the `associatedVoice` property:

\begin{verbatim}
<<
\relative \new Voice = "lahlah" {
  \set Staff.autoBeaming = ##f
  c'4
  \new Voice = "alternative" {
    \voiceOne
    \tuplet 3/2 {
      % show associations clearly.
      \override NoteColumn.force-hshift = #-3
      f8 f g
    }
  }
  \voiceTwo
  f8.[ g16]
  \oneVoice
} >>
a8( b) c
\}
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
\new Lyrics \lyricsto "lahlah" {
  % Tricky: need to set associatedVoice
  % one syllable too soon!
  \set associatedVoice = "alternative" % applies to "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = "lahlah" % applies to "rus"
  sau -- rus Rex
} >>
\end{verbatim}

The text for the first stanza is set to the melody called ‘lahlah’ in the usual way, but the second stanza is set initially to the `lahlah` context and is then switched to the alternative melody for the syllables ‘ran’ to ‘sau’ by the lines:

\begin{verbatim}
\set associatedVoice = "alternative" % applies to "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % applies to "rus"
\end{verbatim}
sau -- rus Rex

Here, alternative is the name of the Voice context containing the triplet.

Note the placement of the \set associatedVoice command – it appears to be one syllable too early, but this is correct.

\begin{center}
\textbf{Note:} The \texttt{\set associatedVoice} command must be placed one syllable \emph{before} the one at which the switch to the new voice is to occur. In other words, changing the associated Voice happens one syllable later than expected. This is for technical reasons, and it is not a bug.
\end{center}

**Printing stanzas at the end**

Sometimes it is appropriate to have one stanza set to the music, and the rest added in verse form at the end of the piece. This can be accomplished by adding the extra verses into a \texttt{\markup} section outside of the main score block. Notice that there are two different ways to force linebreaks when using \texttt{\markup}.

\begin{verbatim}
melody = \relative { 
e' d c d | e e e e |
d d e d | c1 |
}

text = \lyricmode { \set stanza = "1." Ma- ry had a lit- tle lamb, its fleece was white as snow. }

\score { << \new Voice = "one" { \melody } \new Lyrics \lyricsto "one" \text >> \layout { } } \markup { \column { \line { Verse 2. } \line { All the children laughed and played } \line { To see a lamb at school. } } } \markup { \wordwrap-string " Verse 3. Mary took it home again, It was against the rule." }
\end{verbatim}
Verse 2.
All the children laughed and played
To see a lamb at school.

Verse 3.
Mary took it home again,
It was against the rule.

Printing stanzas at the end in multiple columns
When a piece of music has many verses, they are often printed in multiple columns across the page. An outdented verse number often introduces each verse. The following example shows how to produce such output in LilyPond.

```latex
\begin{verbatim}
\set notation = \sratio4 \sratio4
\set textfont = \sratio4 \sratio4
\set textstyle = \sratio4 \sratio4
\set stave = \sratio4 \sratio4
\set staff = \sratio4 \sratio4
\set time = \sratio4 \sratio4

verse = \sratio4 \sratio4
\begin{verbatim}
\set stanza = "1." This is verse one.
It has two lines.
\end{verbatim}
\end{verbatim}
\end{verbatim}
```

```latex
\begin{verbatim}
\set notation = \sratio4 \sratio4
\set textfont = \sratio4 \sratio4
\set textstyle = \sratio4 \sratio4
\set stave = \sratio4 \sratio4
\set staff = \sratio4 \sratio4
\set time = \sratio4 \sratio4

verse = \sratio4 \sratio4
\begin{verbatim}
\set stanza = "1." This is verse one.
It has two lines.
\end{verbatim}
\end{verbatim}
```

```latex
\begin{verbatim}
\set notation = \sratio4 \sratio4
\set textfont = \sratio4 \sratio4
\set textstyle = \sratio4 \sratio4
\set stave = \sratio4 \sratio4
\set staff = \sratio4 \sratio4
\set time = \sratio4 \sratio4

verse = \sratio4 \sratio4
\begin{verbatim}
\set stanza = "1." This is verse one.
It has two lines.
\end{verbatim}
\end{verbatim}
```

```latex
\begin{verbatim}
\set notation = \sratio4 \sratio4
\set textfont = \sratio4 \sratio4
\set textstyle = \sratio4 \sratio4
\set stave = \sratio4 \sratio4
\set staff = \sratio4 \sratio4
\set time = \sratio4 \sratio4

verse = \sratio4 \sratio4
\begin{verbatim}
\set stanza = "1." This is verse one.
It has two lines.
\end{verbatim}
\end{verbatim}
```
2. This is verse two. It has two lines.

3. This is verse three. It has two lines.

4. This is verse four. It has two lines.

5. This is verse five. It has two lines.

See also

Internals Reference: Section “LyricText” in Internals Reference, Section “StanzaNumber” in Internals Reference.

2.1.4 Songs

References for songs

Songs are usually written on three staves with the melody for the singer on the top staff and two staves of piano accompaniment at the bottom. The lyrics of the first stanza are printed immediately underneath the top staff. If there are just a small number of further stanzas these can be printed immediately under the first one, but if there are more stanzas than can be easily accommodated there the second and subsequent stanzas are printed after the music as stand-alone text.

All the notational elements needed to write songs are fully described elsewhere:

- For constructing the staff layout, see Section 1.6.1 [Displaying staves], page 191.
- For writing piano music, see Section 2.2 [Keyboard and other multi-staff instruments], page 338.
- For writing the lyrics to a melody line, see Section 2.1.1 [Common notation for vocal music], page 274.
• For placing the lyrics, see [Placing lyrics vertically], page 289.
• For entering stanzas, see Section 2.1.3 [Stanzas], page 306.
• Songs are frequently printed with the chording indicated by chord names above the staves. This is described in Section 2.7.2 [Displaying chords], page 429.
• To print fret diagrams of the chords for guitar accompaniment or accompaniment by other fretted instruments, see “Fret diagram markups” in Section 2.4.1 [Common notation for fretted strings], page 354.

See also

Notation Reference: Section 2.1.1 [Common notation for vocal music], page 274, Section 2.7.2 [Displaying chords], page 429, Section 1.6.1 [Displaying staves], page 191, Section 2.2 [Keyboard and other multi-staff instruments], page 338, [Placing lyrics vertically], page 289, Section 2.1.3 [Stanzas], page 306.
Snippets: Section “Vocal music” in Snippets.

Lead sheets

Lead sheets may be printed by combining vocal parts and ‘chord mode’; this syntax is explained in Section 2.7 [Chord notation], page 423.

Selected Snippets

Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```
<<
\chords { c2 g:sus4 f e }
\new Staff \relative c' { 
a4 e c8 e r4
b2 c4( d) }
\addlyrics { One day this shall be free __ }
>>
```

![Simple lead sheet example]

See also

Notation Reference: Section 2.7 [Chord notation], page 423.

2.1.5 Choral

This section discusses notation issues that relate most directly to choral music. This includes anthems, part songs, oratorio, etc.

References for choral

Choral music is usually notated on two, three or four staves within a ChoirStaff group. Accompaniment, if required, is placed beneath in a PianoStaff group, which is usually reduced in size for rehearsal of a cappella choral works. The notes for each vocal part are placed in a
Voice context, with each staff being given either a single vocal part (i.e., one Voice) or a pair of vocal parts (i.e., two Voices).

Words are placed in Lyrics contexts, either underneath each corresponding music staff, or one above and one below the music staff if this contains the music for two parts.

Several common topics in choral music are described fully elsewhere:

- An introduction to creating an SATB vocal score can be found in the Learning Manual, see Section “Four-part SATB vocal score” in Learning Manual. There is also a built-in template which simplifies the entry of SATB vocal music, see Section “Built-in templates” in Learning Manual.
- Several templates suitable for various styles of choral music can also be found in the Learning Manual, see Section “Vocal ensembles templates” in Learning Manual.
- For information about ChoirStaff and PianoStaff see [Grouping staves], page 193.
- Shape note heads, as used in Sacred Harp and similar notation, are described in [Shape note heads], page 42.
- When two vocal parts share a staff the stems, ties, slurs, etc., of the higher part will be directed up and those of the lower part down. To do this, use \voiceOne and \voiceTwo. See [Single-staff polyphony], page 173.
- When a vocal part temporarily splits, you should use Temporary polyphonic passages (see [Single-staff polyphony], page 173).

Predefined commands
\oneVoice, \voiceOne, \voiceTwo.

See also

Notation Reference: Section 5.1.7 [Context layout order], page 614, [Grouping staves], page 193, [Shape note heads], page 42, [Single-staff polyphony], page 173.

Snippets: Section “Vocal music” in Snippets.

Internals Reference: Section “ChoirStaff” in Internals Reference, Section “Lyrics” in Internals Reference, Section “PianoStaff” in Internals Reference.

Score layouts for choral
Choral music containing four staves, with or without piano accompaniment, is usually laid out with two systems per page. Depending on the page size, achieving this may require changes to several default settings. The following settings should be considered:

- The global staff size can be modified to change the overall size of the elements of the score. See Section 4.2.2 [Setting the staff size], page 559.
- The distances between the systems, the staves and the lyrics can all be adjusted independently. See Section 4.4 [Vertical spacing], page 566.
- The dimensions of the vertical layout variables can be displayed as an aid to adjusting the vertical spacing. This and other possibilities for fitting the music onto fewer pages are described in Section 4.6 [Fitting music onto fewer pages], page 593.
- If the number of systems per page changes from one to two it is customary to indicate this with a system separator mark between the two systems. See [Separating systems], page 198.
- For details of other page formatting properties, see Section 4.1 [Page layout], page 546.
Dynamic markings by default are placed below the staff, but in choral music they are usually placed above the staff in order to avoid the lyrics. The predefined command \texttt{\texttt{dynamicUp}} does this for the dynamic markings in a single \texttt{Voice} context. If there are many \texttt{Voice} contexts this predefined command would have to be placed in every one. Alternatively its expanded form can be used to place all dynamic markings in the entire score above their respective staves, as shown here:

\begin{verbatim}
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative { g'4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
        \relative { d'4 d d\p d }
      }
    }
>>
  \layout {
    \context {
      \Score
      \override DynamicText.direction = #UP
      \override DynamicLineSpanner.direction = #UP
    }
  }
}
\end{verbatim}

\begin{verbatim}
\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{exampleImage.png}
\caption{Example of dynamic markings placed above the staff.}
\end{figure}
\end{verbatim}

	extbf{Predefined commands}

\texttt{\texttt{dynamicUp}}, \texttt{\texttt{dynamicDown}}, \texttt{\texttt{dynamicNeutral}}.

\textbf{See also}

Notation Reference: Section 4.6.2 [Changing spacing], page 595, Section 4.6.1 [Displaying spacing], page 594, Section 4.6 [Fitting music onto fewer pages], page 593, Section 4.1 [Page layout], page 546, Section 4.2 [Score layout], page 556, [Separating systems], page 198, Section 4.2.2 [Setting the staff size], page 559, Section 4.3 [Breaks], page 560, Section 4.4 [Vertical spacing], page 566.

Selected Snippets

*Using arpeggioBracket to make divisi more visible*

The `arpeggioBracket` can be used to indicate the division of voices where there are no stems to provide the information. This is often seen in choral music.

```latex
\include "english.ly"

\score {
  \relative c' { 
    \key a \major 
    \time 2/2 
    \new Voice = "upper"
    \{ \voiceOne \arpeggioBracket 
      a2( b2
      <b d>1\arpeggio)
      <cs e>\arpeggio ~
      <cs e>4
    \}
    \addlyrics { \lyricmode { A -- men. } } 
  } 
  \new Voice = "lower"
  \{ \voiceTwo 
    a1 ~
    a ~
    a4 \bar "."
  } 
  \} 
  \layout { ragged-right = ##t }
}
```

See also

Notation Reference: Section 1.3.3 [Expressive marks as lines], page 140.

**2.1.6 Opera and stage musicals**

The music, lyrics and dialogue to opera and stage musicals are usually set out in one or more of the following forms:

- A *Conductors’ Score* containing the full orchestral and vocal parts, together with libretto cues if there are spoken passages.
- *Orchestral Parts* containing the music for the individual instruments of the orchestra or band.
A **Vocal Score** containing all vocal parts with piano accompaniment. The accompaniment is usually an orchestral reduction, and if so the name of the original orchestral instrument is often indicated. Vocal scores sometimes includes stage directions and libretto cues.

A **Vocal Book** containing just the vocal parts (no accompaniment), sometimes combined with the libretto.

A **Libretto** containing the extended passages of spoken dialogue usually found in musicals, together with the words to the sung parts. Stage directions are usually included. LilyPond can be used to typeset libretti but as they contain no music alternative methods may be preferable.

The sections in the LilyPond documentation which cover the topics needed to create scores in the styles commonly found in opera and musicals are indicated in the References below. This is followed by sections covering those techniques which are peculiar to typesetting opera and musical scores.

### References for opera and stage musicals

In addition to vocal and stage ensembles, most of the following notions may apply to nearly any orchestral and ensemble music:

- A conductors’ score contains many grouped staves and lyrics. Ways of grouping staves is shown in [Grouping staves], page 193. To nest groups of staves see [Nested staff groups], page 196.
- The printing of empty staves in conductors’ scores and vocal scores is often suppressed. To create such a “Frenched score” see [Hiding staves], page 205.
- Writing orchestral parts is covered in Section 1.6.3 [Writing parts], page 210. Other sections in the Specialist notation chapter may be relevant, depending on the orchestration used. Many instruments are transposing instruments, see [Instrument transpositions], page 27.
- If the number of systems per page changes from page to page it is customary to separate the systems with a system separator mark. See [Separating systems], page 198.
- For details of other page formatting properties, see Section 4.1 [Page layout], page 546.
- Dialogue cues, stage directions and footnotes can be inserted, see Section 3.2.4 [Creating footnotes], page 505, and Section 1.8 [Text], page 242. Extensive stage directions can also be added with a section of stand-alone markups between two \score blocks, see [Separate text], page 251.

### See also


Notation Reference: Section 3.2.4 [Creating footnotes], page 505, [Grouping staves], page 193, [Hiding staves], page 205, [Instrument transpositions], page 27, [Nested staff groups], page 196, Section 4.1 [Page layout], page 546, [Separating systems], page 198, [Transpose], page 11, Section 1.6.3 [Writing parts], page 210, Section 1.8.1 [Writing text], page 243.

Snippets: Section “Vocal music” in *Snippets*.

### Character names

Character names are usually shown to the left of the staff when the staff is dedicated to that character alone:

```lilypond
\score {
  <<
    \new Staff {
```
When two or more characters share a staff the character’s name is usually printed above the staff at the start of every section applying to that character. This can be done with markup. Often a specific font is used for this purpose.
Alternatively, if there are many character changes, it may be easier to set up variables to hold the definitions for each character so that the switch of characters can be indicated easily and concisely.

```latex
kaspar = {
\clef "G_8"
\set Staff.shortVocalName = "Kas."
\set Staff.midiInstrument = "voice oohs"
<>\markup \smallCaps "Kaspar"
}

melchior = {
\clef "bass"
\set Staff.shortVocalName = "Mel."
\set Staff.midiInstrument = "choir aahs"
<>\markup \smallCaps "Melchior"
}

\relative c' {
\kaspar
\set Staff.midiInstrument = "voice oohs"
\set Staff.shortVocalName = "Kas."
\melchior
\set Staff.midiInstrument = "choir aahs"
\set Staff.shortVocalName = "Mel."
\kaspar
\melchior
\kaspar
\melchior
\kaspar
```

See also

- Notation Reference: Section 1.8 [Text], page 242, Section A.11 [Text markup commands], page 711.

Musical cues

Musical cues can be inserted in Vocal Scores, Vocal Books and Orchestral Parts to indicate what music in another part immediately precedes an entry. Also, cues are often inserted in the piano reduction in Vocal Scores to indicate what each orchestral instrument is playing. This aids the conductor when a full Conductors’ Score is not available.

The basic mechanism for inserting cues is fully explained in the main text, see [Quoting other voices], page 214, and [Formatting cue notes], page 217. But when many cues have to be inserted, for example, as an aid to a conductor in a vocal score, the instrument name must be positioned carefully just before and close to the start of the cue notes. The following example shows how this is done.

```latex
flute = \relative {
\set Staff.midiInstrument = "flute"
\set Staff.shortVocalName = "Fl."\set Staff.midiInstrument = "flute"
\relative c' {
\flute
s4 s4 e'' g
```

```
If a transposing instrument is being quoted the instrument part should specify its key so the conversion of its cue notes will be done automatically. The example below shows this transposition for a B-flat clarinet. The notes in this example are low on the staff so \texttt{DOWN} is specified in \texttt{\cueDuring} (so the stems are down) and the instrument name is positioned below the staff.

\begin{verbatim}
clarinet = \relative c' { 
  \transposition bes 
  fis4 d d c 
}
\addQuote "clarinet" { \clarinet }
\end{verbatim}

\begin{verbatim}
pianoRH = \relative c'' { 
  \transposition c' 
  \% position name of cue-ing instrument below the staff 
  <>\markup { \right-align { \tiny "Clar." } } 
  \cueDuring "clarinet" #DOWN { c4. g8 } 
  g4 bes4 
}
pianoLH = \relative { c4 <c' e> e, <g c> }
\end{verbatim}
From these two examples it is clear that inserting many cues in a Vocal Score would be tedious, and the notes of the piano part would become obscured. However, as the following snippet shows, it is possible to define a music function to reduce the amount of typing and to make the piano notes clearer.

### Selected Snippets

**Adding orchestral cues to a vocal score**

This shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function \cuesWhile takes four arguments: the music from which the cue is to be taken, as defined by \addQuote, the name to be inserted before the cue notes, then either \UP or \DOWN to specify either \voiceOne with the name above the staff or \voiceTwo with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

\cuesWhile =

\{(define-music-function
  (instrument name dir music)
  (string? string? ly:dir? ly:music?)
  #\{
    \cuesDuring $instrument #dir {
      \once \override TextScript.self-alignment-X = #RIGHT
      \once \override TextScript.direction = $dir
      <-\markup { \tiny #name }
      $music
    }
  #})

flute = \relative c'' {

\transposition c' 
\relative c' {
  s4 s4 e g
}
\addQuote "flute" { \flute }

clarinet = \relative c' {
  \transposition bes 
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }

words = \lyricmode { here's the lyrics }

pianoRH = \relative c'' { 
  \transposition c' 
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 } 
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}

pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" { 
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
See also


Notation Reference: Section 5.5.1 [Aligning objects], page 650, Section 5.4.2 [Direction and placement], page 634, [Formatting cue notes], page 217, [Quoting other voices], page 214, Section 5.6 [Using music functions], page 662.

Snippets: Section “Vocal music” in Snippets.

Internals Reference: Section “CueVoice” in Internals Reference.

Known issues and warnings
\cueDuring automatically inserts a CueVoice context and all cue notes are placed in that context. This means it is not possible to have two overlapping sequences of cue notes by this technique. Overlapping sequences could be entered by explicitly declaring separate CueVoice contexts and using \quoteDuring to extract and insert the cue notes.

Spoken music

Such effects as ‘parlato’ or ‘Sprechgesang’ require performers to speak without pitch but still with rhythm; these are notated by cross note heads, as demonstrated in [Special note heads], page 39.

Dialogue over music

Dialogue over music is usually printed over the staves in an italic font, with the start of each phrase keyed in to a particular music moment.

For short interjections a simple markup suffices.

\relative {
  a'4\markup { \smallCaps { Alex - } \italic { He’s gone } } a a a 
  a4 a a\markup { \smallCaps { Bethan - } \italic Where? } a 
  a4 a a a }

ALEX - He’s gone  BETHAN - Where?

For longer phrases it may be necessary to expand the music to make the words fit neatly. There is no provision in LilyPond to do this fully automatically, and some manual intervention to layout the page will be necessary.

For long phrases or for passages with a lot of closely packed dialogue, using a Lyrics context will give better results. The Lyrics context should not be associated with a music Voice; instead each section of dialogue should be given an explicit duration. If there is a gap in the dialogue,
the final word should be separated from the rest and the duration split between them so that the underlying music spaces out smoothly.

If the dialogue extends for more than one line it will be necessary to manually insert \breaks and adjust the placing of the dialogue to avoid running into the right margin. The final word of the last measure on a line should also be separated out, as above.

Here is an example illustrating how this might be done.

```latex
\begin{verbatim}
music = \relative { 
    \repeat unfold 3 { a'4 a a a }
}

dialogue = \lyricmode {
    \markup {
        \fontsize #1 \upright \smallCaps Abe: 
        "Say this over measures one and"
    }4*7
    "two"4 |
    \break
    "and this over measure"4*3
    "three"4 |
}
\end{verbatim}
```

See also

Notation Reference: [Manual syllable durations], page 280, Section 1.8 [Text], page 242.
Internal Reference: Section “LyricText” in Internals Reference.
2.1.7 Chants, psalms and hymns

The music and words for chants, psalms and hymns usually follow a well-established format in any particular church. Although the formats may differ from church to church the type-setting problems which arise are broadly similar, and are covered in this section.

References for chants and psalms

Typesetting Gregorian chant in various styles of ancient notation is described in Section 2.9 [Ancient notation], page 446.

See also

Notation reference: Section 2.9 [Ancient notation], page 446.
Snippets: Section “Vocal music” in Snippets.

Setting a chant

Modern chant settings use modern notation with varying numbers of elements taken from ancient notation. Some of the elements and methods to consider are shown here.

Chants often use quarter notes without stems to indicate the pitch, with the rhythm being taken from the spoken rhythm of the words.

\stemOff = { \hide Staff.Stem }

\relative c' { \stemOff a'4 b c2 | }

Chants often omit the bar lines or use shortened or dotted bar lines to indicate pauses in the music. To omit all bar lines from all staves remove the bar line engraver completely:

\score {
  \new StaffGroup <<
    \new Staff { \relative { a'4 b c2 | a4 b c2 | a4 b c2 | } }
    \new Staff { \relative { a'4 b c2 | a4 b c2 | a4 b c2 | } }
  >>
  \layout { }
Bar lines can also be removed on a staff-by-staff basis:

To remove bar lines from just a section of music treat it as a cadenza. If the section is long you may need to insert dummy bar lines with `\bar ""` to show where the line should break.
Rests or pauses in chants can be indicated by modified bar lines.

```
\relative a' {  
a4  
\cadenzaOn  
b c2  
a4 b c2  
\bar "-"  
a4 b c2  
a4 b c2  
\bar ";"  
a4 b c2  
\bar "!"  
a4 b c2  
\bar "||"  
}
```

Alternatively, the notation used in Gregorian chant for pauses or rests is sometimes used even though the rest of the notation is modern. This uses a modified \breathe mark:

```
divisioMinima = {  
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-minima  
  \once \override BreathingSign.Y-offset = #0  
  \breathe  
}  
divisioMaior = {  
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maior  
  \once \override BreathingSign.Y-offset = #0  
  \breathe  
}  
divisioMaxima = {  
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maxima  
  \once \override BreathingSign.Y-offset = #0  
  \breathe  
}  
finalis = {  
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis  
  \once \override BreathingSign.Y-offset = #0  
  \breathe  
}
```
Chants usually omit the time signature and often omit the clef too.

Chants for psalms in the Anglican tradition are usually either *single*, with 7 bars of music, or *double*, with two lots of 7 bars. Each group of 7 bars is divided into two halves, corresponding to the two halves of each verse, usually separated by a double bar line. Only whole and half notes are used. The 1st bar in each half always contains a single chord of whole notes. This is the “reciting note”. Chants are usually centered on the page.
\begin{verbatim}
g'1 | c2 b | a1 \bar "||"
  a1 | d2 c | c b | c1 \bar "||"
\}

AltoMusic = \relative {
e'1 | g2 g | f1 |
f1 | f2 e | d d | e1 |
}

TenorMusic = \relative {
c'1 | c2 c | c1 |
d1 | g,2 g | g g | g1 |
}

BassMusic = \relative {
c1 | e2 e | f1 |
d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score { % centered
      <<
        \new ChoirStaff <<
        \new Staff <<
          \global
          \clef "treble"
          \new Voice = "Soprano" <<
            \voiceOne
            \SopranoMusic
          >>
        \new Voice = "Alto" <<
          \voiceTwo
          \AltoMusic
        >>
      >>
      \new Staff <<
        \clef "bass"
        \global
        \new Voice = "Tenor" <<
          \voiceOne
          \TenorMusic
        >>
        \new Voice = "Bass" <<
          \voiceTwo
          \BassMusic
        >>
    >>
  } % centered
} % fill-line
\end{verbatim}
Some other approaches to setting such a chant are shown in the first of the following snippets.

**Selected Snippets**

*Chant or psalms notation*

This form of notation is used for Psalm chant, where verses aren’t always the same length.

```latex
\stemOff = \hide Staff.Stem
\stemOn = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve\markup { \italic flexe } \mark up
    \stemOn g'2 \bar "||"
  }
}
```

```
\markup { flexe }
```

```latex
\layout { 
  \context { 
    \Score 
    \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/2) 
  } 
  \context { 
    \Staff 
    \remove "Time_signature_engraver"
  } 
} % End score

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
   {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve\markup { \italic flexe } \mark up
    \stemOn g'2 \bar "||"
  }
}
```

```
\markup { flexe }
```
Canticles and other liturgical texts may be set more freely, and may use notational elements from ancient music. Often the words are shown underneath and aligned with the notes. If so, the notes are spaced in accordance with the syllables rather than the notes' durations.

Ancient notation template – modern transcription of Gregorian music

This example demonstrates how to do modern transcription of Gregorian music. Gregorian music has no measure, no stems; it uses only half and quarter note heads, and special marks, indicating rests of different length.

\include "gregorian.ly"

chant = \relative c' { \
\set Score.timing = ##f
f4 a2 \divisioMinima
G4 B a2 f2 \divisioMaior
G4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
\new Staff <<
  \new Voice = "melody" chant
  \new Lyrics = "one" lyricsto melody verba
>>
\layout {
  \context {
    \Staff
    \remove "Time_signature_engraver"
    \remove "Bar_engraver"
    \hide Stem
  }
  \context {
    \Voice
    \override Stem.length = #0
  }
  \context {
    \Score
    barAlways = ##t
  }
}
}

Lorem ipsum dolor sit a-met

See also

Pointing a psalm

The words to an Anglican psalm are usually printed in separate verses centered underneath the chant.

Single chants (with 7 bars) are repeated for every verse. Double chants (with 14 bars) are repeated for every pair of verses. Marks are inserted in the words to show how they should be fitted to the chant. Each verse is divided into two halves. A colon is usually used to indicate this division. This corresponds to the double bar line in the music. The words before the colon are sung to the first three bars of music; the words after the colon are sung to the last four bars.

Single bar lines (or in some psalters an inverted comma or similar symbol) are inserted between words to indicate where the bar lines in the music fall. In markup mode a single bar line can be entered with the bar check symbol, \ |

\markup{
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing | unto the | Lord : let }
        \line { us heartily rejoice in the | strength of | our }
        \line { sal- | -vation. }
      }
    }
  }
}

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Other symbols may require glyphs from the fetaMusic fonts. For details, see Section 1.8.3 [Fonts], page 267.

tick = \markup{
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup{
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing \tick unto the \tick Lord : let }
        \line { us heartily rejoice in the \tick strength of \tick our }
        \line { sal \tick vation. }
      }
    }
  }
}
O come let us sing ‘unto the ’Lord : let
us heartily rejoice in the ’strength of’our
sal’vation.

Where there is one whole note in a bar all the words corresponding to that bar are recited on
that one note in speech rhythm. Where there are two notes in a bar there will usually be only
one or two corresponding syllables. If there are more than two syllables a dot is usually inserted
to indicate where the change in note occurs.

\begin{music}
\markup {
\fill-line {
\column {
\left-align {
\line { O come let us sing \tick unto \dot the \tick Lord : let }
\line { us heartily rejoice in the \tick strength of \tick our }
\line { sal \tick vation. }
}
}\}
}

O come let us sing ‘unto the ’Lord : let
us heartily rejoice in the ’strength of’our
sal’vation.

In some psalters an asterisk is used to indicate a break in a recited section instead of a
comma, and stressed or slightly lengthened syllables are indicated in bold text.

\begin{music}
\markup {
\fill-line {
\column {
\left-align {
\line { Today if ye will hear his voice * }
\line { not your | hearts : as in the pro-
}\line { vocation * and as in the \bold day of tempt- |
\line { -ation | in the \wilderness. }
}
}
\end{music}
Today if ye will hear his voice *

**harden** | not your | hearts : as in the pro-
vocation * and as in the **day** of tempt- |
-ation | in the | wilderness.

In other psalters an accent is placed over the syllable to indicate stress.

```
tick = \markup {
    \raise #2 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
```

```
\markup {
    \fill-line {
        \column {
            \left-align {
                \line {
                    O come let us \concat {
                        si \combine \tick ng
                    }
                    | unto the | Lord : let
                }
                \line {
                    us heartily \concat {
                        rejo \combine \tick ice
                    }
                    in the | strength of | our
                }
                \line { sal- | -vation. }
            }
        }
    }
}
```

```
O come let us **sing** | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.
```

The use of markup to center text, and arrange lines in columns is described in Section 1.8.2 [Formatting text], page 252.

Most of these elements are shown in one or other of the two verses in the template, see Section “Psalms” in Learning Manual.

**See also**


Notation Reference: Section 1.8.3 [Fonts], page 267, Section 1.8.2 [Formatting text], page 252.
Partial measures in hymn tunes

Hymn tunes frequently start and end every line of music with partial measures so that each line of music corresponds exactly with a line of text. This requires a \texttt{\partial} command at the start of the music and \texttt{\bar "|"} or \texttt{\bar "||"} commands at the end of each line.

\textit{Hymn template}

This code shows one way of setting out a hymn tune when each line starts and ends with a partial measure. It also shows how to add the verses as stand-alone text under the music.

\begin{verbatim}
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic = \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

global = {
  \key g \major
}

\score { % Start score
  \new PianoStaff << % Start pianostaff
  \new Staff << % Start Staff = RH
  \global
  \clef "treble"
  \new Voice = "Soprano" << % Start Voice = "Soprano"
    Timeline
    voiceOne
    SopranoMusic
  >> % End Voice = "Soprano"
  \new Voice = "Alto" << % Start Voice = "Alto"
}\
\end{verbatim}
\Timeline
\voiceTwo
\AltoMusic
>> % End Voice = "Alto"
>> % End Staff = RH
\new Staff << % Start Staff = LH
\global
\clef "bass"
\new Voice = "Tenor" << % Start Voice = "Tenor"
\Timeline
\voiceOne
\TenorMusic
>> % End Voice = "Tenor"
\new Voice = "Bass" << % Start Voice = "Bass"
\Timeline
\voiceTwo
\BassMusic
>> % End Voice = "Bass"
>> % End Staff = LH
>> % End pianostaff

\markup {
  \fill-line {
    ""
    \column {
      \left-align {
        "This is line one of the first verse"
        "This is line two of the same"
        "And here's line three of the first verse"
        "And the last line of the same"
      }
    }
  }
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music
} % End paper block
This is line one of the first verse
This is line two of the same
And here's line three of the first verse
And the last line of the same

2.1.8 Ancient vocal music
Ancient vocal music is supported, as explained in Section 2.9 [Ancient notation], page 446.

See also
Notation Reference: Section 2.9 [Ancient notation], page 446.

2.2 Keyboard and other multi-staff instruments
This section discusses several aspects of music notation that are unique to keyboard instruments and other instruments notated on many staves, such as harps and vibraphones. For the purposes of this section this entire group of multi-staff instruments is called “keyboards” for short, even though some of them do not have a keyboard.

2.2.1 Common notation for keyboards

This section discusses notation issues that may arise for most keyboard instruments.

References for keyboards

Keyboard instruments are usually notated with Piano staves. These are two or more normal staves coupled with a brace. The same notation is also used for other keyed instruments. Organ music is normally written with two staves inside a PianoStaff group and third, normal staff for the pedals.

The staves in keyboard music are largely independent, but sometimes voices can cross between the two staves. This section discusses notation techniques particular to keyboard music.

Several common issues in keyboard music are covered elsewhere:

- Keyboard music usually contains multiple voices and the number of voices may change regularly; this is described in [Collision resolution], page 178.
- Keyboard music can be written in parallel, as described in [Writing music in parallel], page 188.
- Dynamics may be placed in a Dynamics context, between the two Staff contexts to align the dynamic marks on a horizontal line centered between the staves; see [Dynamics], page 125.
- Fingerings are indicated with [Fingering instructions], page 229.
- Organ pedal indications are inserted as articulations, see Section A.14 [List of articulations], page 768.
- Vertical grid lines can be shown with [Grid lines], page 238.
- Keyboard music often contains Laissez vibrer ties as well as ties on arpeggios and tremolos, described in [Ties], page 55.
- Placing arpeggios across multiple voices and staves is covered in [Arpeggio], page 145.
- Tremolo marks are described in [Tremolo repeats], page 166.
- Several of the tweaks that can occur in keyboard music are demonstrated in Section “Real music example” in Learning Manual.
- Hidden notes can be used to produce ties that cross voices, as shown in Section “Other uses for tweaks” in Learning Manual.

See also

Notation Reference: [Grouping staves], page 193, [Instrument names], page 210, [Collision resolution], page 178, [Writing music in parallel], page 188, [Fingering instructions], page 229, Section A.14 [List of articulations], page 768, [Grid lines], page 238, [Ties], page 55, [Arpeggio], page 145, [Tremolo repeats], page 166.

Internals Reference: Section “PianoStaff” in Internals Reference.

Snippets: Section “Keyboards” in Snippets.

Changing staff manually

Voices can be switched between staves manually, using the command

\change Staff = staffname

The string staffname is the name of the staff. It switches the current voice from its current staff to the staff called staffname. Typical values for staffname are "up" and "down", or "RH" and "LH".

The staff to which the voice is being switched must exist at the time of the switch. If necessary, staves should be “kept alive”, see Section 5.1.3 [Keeping contexts alive], page 602, or explicitly instantiated, for example by using the empty chord, <>, see [Chorded notes], page 168.

\new PianoStaff <<
\new Staff = "up" {
  % enforce creation of all contexts at this point of time
  <>
  \change Staff = "down" c2
  \change Staff = "up" c'2
}
\new Staff = "down" {
  clef bass
  % keep staff alive
  s1
}
>>

Cross-staff notes are beamed automatically:

\new PianoStaff <<
\new Staff = "up" {
  <e' c'>8
  \change Staff = "down"
  g8 fis g
  \change Staff = "up"
  <g' c''>8
  \change Staff = "down"
  e8 dis e
  \change Staff = "up"
}
\new Staff = "down" {

If the beaming needs to be tweaked, make any changes to the stem directions first. The beam positions are then measured from the center of the staff that is closest to the beam. For a simple example of beam tweaking, see Section “Fixing overlapping notation” in Learning Manual.

Overlapping notation can result when voices cross staves:

\new PianoStaff <<
\new Staff = "up" {
    \voiceOne
    \percent Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner.staff-padding = #4
e' '2\p\< d' '\>
c' '1\! 
}
\new Staff = "down" <<
{
    \clef bass
    s4. e,8\rest g,2\rest
c1
} \\ {
    c8\( g c' \\
    \change Staff = "up"
    e' g' b'-3 a' g'\)
f'1 
}
>>

The stem and slur overlap the intervening line of dynamics because automatic collision resolution is suspended for beams, slurs and other spanners that connect notes on different staves, as well as for stems and articulations if their placement is affected by a cross-staff spanner. The resulting collisions must be resolved manually, where necessary, using the methods in Section “Fixing overlapping notation” in Learning Manual.
See also


Notation Reference: [Stems], page 234, [Automatic beams], page 84, Section 5.1.3 [Keeping contexts alive], page 602.

Snippets: Section “Keyboards” in Snippets.


Known issues and warnings

Beam collision avoidance does not work for automatic beams that end right before a change in staff. In this case use manual beams.

Changing staff automatically

Voices can be made to switch automatically between the top and the bottom staff. The syntax for this is

```
\autoChange ...music...
```

This will create two staves inside the current staff group (usually a PianoStaff), called "up" and "down". The lower staff will be in the bass clef by default. The autoChanger switches on the basis of the pitch (middle C is the turning point), and it looks ahead skipping over rests to switch in advance.

```
\new PianoStaff {
  \autoChange {
    g4 a b c'
    d'4 r a g
  }
}
```

It is possible to specify other pitches for the turning point. If the staves are not instantiated explicitly, other clefs may be used.

```
music = {
  g8 b a c' b8 d' c'8 e'
  d'8 r f' g' a'2
}
```

```
\autoChange d' \music
\autoChange b \with { \clef soprano } \music
\autoChange d' \with { \clef alto } \with { \clef tenor } \music
```
A `\relative` section that is outside of `\autoChange` has no effect on the pitches of the music, so if necessary, put `\relative` inside `\autoChange`.

If additional control is needed over the individual staves, they can be created manually with the names "up" and "down". The `\autoChange` command will then switch its voice between the existing staves.

**Note:** If staves are created manually, they must be named "up" and "down".

For example, staves must be created manually in order to place a key signature in the lower staff:

```latex
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melOne" {
      \key g \major
      \autoChange \relative {
        g8 b a c b d c e
d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```

See also

Notation Reference: [Changing staff manually], page 340.
Snippets: Section “Keyboards” in Snippets.
Known issues and warnings

The staff switches may not end up in optimal places. For high quality output, staff switches should be specified manually.

Chords will not be split across the staves; they will be assigned to a staff based on the first note named in the chord construct.

Staff-change lines

Whenever a voice switches to another staff, a line connecting the notes can be printed automatically:

\new PianoStaff <<
\new Staff = "one" {
    \showStaffSwitch
c'1
    \change Staff = "two"
b2 a
}
\new Staff = "two" {
    \clef bass
    s1*2
}
>>

A staff-change line between chords connects the chords’ “last notes” as written in the source file; this can be used to quickly adjust the line’s vertical start and end positions.

\new PianoStaff <<
\new Staff = "one" {
    <c' e' g'>1
    \showStaffSwitch
    \change Staff = "two"
    <a c' f>1
    \hideStaffSwitch
    \change Staff = "one"
    <e' g' c'>1
    \showStaffSwitch
    \change Staff = "two"
    <f a c'>1
}
\new Staff = "two" {
    \clef bass
    s1*4
}
>>
Predefined commands
\showStaffSwitch, \hideStaffSwitch.

See also
Snippets: Section “Keyboards” in Snippets.
Internals Reference: Section “Note_head_line_engraver” in Internals Reference, Section “VoiceFollower” in Internals Reference.

Selected Snippets
Cross staff stems
This snippet shows the use of the Span_stem_engraver and \crossStaff to connect stems across staves automatically.

The stem length need not be specified, as the variable distance between noteheads and staves is calculated automatically.

\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{\new PianoStaff <<
  \new Staff {
    <b d’>4 r d’16\> e’8. g8 r! 
    e’8 f’ g’4 e’2
  }
  \new Staff {
    \clef bass 
    \voiceOne
    \autoBeamOff
    \crossStaff { <e g>4 e, g16 a8. c8} d 
    \autoBeamOn 
    g8 f g4 c2
  }
  >>
}


For the time being, this engraver can not be specified by its name in double quotes. Instead its name must be prefixed with a hash symbol (#) due to the way it is implemented.

*Indicating cross-staff chords with arpeggio bracket*

An arpeggio bracket can indicate that notes on two different staves are to be played with the same hand. In order to do this, the PianoStaff must be set to accept cross-staff arpeggios and the arpeggios must be set to the bracket shape in the PianoStaff context.

(Debussy, Les collines d’Anacapri, m. 65)

```latex
\new PianoStaff <<
\set PianoStaff.connectArpeggios = ##t
\override PianoStaff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
\new Staff {
    \relative c' {
        \key b \major
        \time 6/8
        b8-.\arpeggio fis'-.\> cis-. e-. gis-. b-.\}!\fermata\laissezVibrer
    \bar "||"
}
} \new Staff {
    \relative c' {
        \clef bass
        \key b \major
        <<
        { <a e cis>2.\arpeggio }
        \\{ <a, e a,>2. } >>
    }
} >>
```

See also

Snippets: Section “Keyboards” in Snippets.
Internals Reference: Section “Stem” in Internals Reference.

**2.2.2 Piano**

This section discusses notation issues that relate most directly to the piano.
Chapter 2: Specialist notation

Piano pedals

Pianos generally have three pedals that alter the way sound is produced: sustain, sostenuto (sos.), and una corda (U.C.). Sustain pedals are also found on vibraphones and celestas.

There are three styles of pedal indications: text, bracket, and mixed. The sustain pedal and the una corda pedal use the text style by default while the sostenuto pedal uses mixed by default.

The placement of the pedal commands matches the physical movement of the sustain pedal during piano performance. Pedalling to the final bar line is indicated by omitting the final pedal off command.

Pedal indications may be placed in a Dynamics context, which aligns them on a horizontal line.

See also

Notation Reference: [Ties], page 55.
Snippets: Section “Keyboards” in Snippets.
2.2.3 Accordian

This section discusses notation that is unique to the accordion.

Discant symbols

Accordions are often built with more than one set of reeds that may be in unison with, an octave above, or an octave below the written pitch. Each accordion maker has different names for the shifts that select the various reed combinations, such as oboe, musette, or bandonium, so a system of symbols has come into use to simplify the performance instructions.

A complete list of all available accordion registers can be found in Section A.11.6 [Accordion Registers], page 753.

Selected Snippets

Accordion register symbols

Accordion register symbols are available as \markup as well as as standalone music events (as register changes tend to occur between actual music events). Bass registers are not overly standardized. The available commands can be found in 'Discant symbols' in the Notation Reference.

```latex
\ #(use-modules (scm accreg))
\new PianoStaff
\new Staff \relative {\clef treble \discant "10" \r 8 s32 \f' [ bes f] \s e[ a e] \s d[ g d] s16 e32[ a]
\new Staff \relative {\clef treble \freeBass "1" \r 8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
\clef bass \stdBass "Master"
\new Staff \relative {\clef treble \freeBass "1" \r 8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
\clef bass \stdBass "Master"
```
See also

Snippets: Section “Keyboards” in Snippets.

2.2.4 Harp
This section discusses notation issues that are unique to the harp.

References for harps
Some common characteristics of harp music are covered elsewhere:

- The glissando is the most characteristic harp technique, [Glissando], page 140.
- A bisbigliando is written as a tremolo [Tremolo repeats], page 166.
- Natural harmonics are covered under [Harmonics], page 352.
- For directional arpeggios and non-arpeggios, see [Arpeggio], page 145.

See also

Notation Reference: [Tremolo repeats], page 166, [Glissando], page 140, [Arpeggio], page 145, [Harmonics], page 352.

Harp pedals
Harps have seven strings per octave that may be sounded at the natural, flattened, or sharpened pitch. In lever harps, each string is adjusted individually, but in pedal harps every string with the same pitch name is controlled by a single pedal. From the player’s left to right, the pedals are D, C, and B on the left and E, F, G, and A on the right. The position of the pedals may be indicated with text marks:

```
cis'\markup { \harp-pedal "v--|vv--" }
c'\markup { \harp-pedal "o--|vv--" }
```

or pedal diagrams:

```
c'\markup { \harp-pedal "-v-|vv-" }
```

or pedal diagrams:
The \texttt{harp-pedal} command accepts a string of characters, where \wedge is the highest pedal position (flattened pitch), \_ is the middle pedal position (natural pitch), \texttt{v} is the lowest pedal position (sharpened pitch), and | is the divider. A prefixed o will circle the following pedal symbol.

See also

Notation Reference: [Text scripts], page 245, Section A.11.5 [Instrument Specific Markup], page 750.

2.3 Unfretted string instruments

This section provides information and references which are helpful when writing for unfretted string instruments, principally orchestral strings.

2.3.1 Common notation for unfretted strings

There is little specialist notation for unfretted string instruments. The music is notated on a single staff, and usually only a single voice is required. Two voices might be required for some double-stopped or divisi passages.
References for unfretted strings

Most of the notation which is useful for orchestral strings and other bowed instruments is covered elsewhere:

- Textual indications such as “pizz.” and “arco” are added as simple text – see [Text scripts], page 245.
- Fingerings, including the thumb indication, are described in [Fingering instructions], page 229.
- String numbers may be added (generally in roman numbers for bowed instruments) as explained in [String number indications], page 354.
- Double stopping is normally indicated by writing a chord, see [Chorded notes], page 168. Directives for playing chords may be added, see [Arpeggio], page 145.
- Templates for string quartets can be found in Section “String quartet templates” in Learning Manual. Others are shown in the snippets.

See also


Notation Reference: [Text scripts], page 245, [Fingering instructions], page 229, [Chorded notes], page 168, [Arpeggio], page 145.

Snippets: Section “Unfretted strings” in Snippets.

Bowing indications

Bowing indications are created as articulations, which are described in [Articulations and ornamentations], page 122.

The bowing commands, \upbow and \downbow, are used with slurs as follows:
\[\relative { c''4(\downbow d) e(\upbow f) }\]

Roman numerals can be used for string numbers (rather than the default circled Arabic numbers), as explained in [String number indications], page 354.

Alternatively, string indications may be printed using markup commands; articulation scripts may also indicate open strings.
\[a'4 \open \romanStringNumbers a'2\]
\[a'2\markup { \small "sul A" }\]

Predefined commands

\downbow, \upbow, \open, \romanStringNumbers.
Harmonics

Natural harmonics

Natural harmonics can be notated in several ways. A diamond-shaped note head generally means to touch the string where you would stop the note if it were not a diamond.

```
relative d'' { 
    d4 e4.
    \harmonicsOn
    d8 e e
    \harmonicsOff
    d4 e4.
    \harmonicsOff
    d8 e e
}
```

Alternatively a normal note head is shown at the pitch to be sounded together with a small circle to indicate it should be played as a harmonic:

```
d''2\flageolet d''_\flageolet
```

A smaller circle may be created, see the snippet list in [References for unfretted strings], page 351.

Artificial harmonics

Artificial harmonics are notated with two notes, one with a normal note head indicating the stopped position and one with an open diamond note head to indicate the harmonic position.

Artificial harmonics indicated with \harmonic do not show the dots. The context property harmonicDots should be set if dots are required.

```
relative e' { 
    <e a\harmonic>2.  <c g'\harmonic>4
    \set harmonicDots = ##t
    <e a\harmonic>2.  <c g'\harmonic>4
}
```

See also


Notation Reference: [Special note heads], page 39, [References for unfretted strings], page 351.
Snap (Bartók) pizzicato

A snap pizzicato (also known as “Bartok pizz”) is a type of pizzicato where the string is deliberately plucked upwards (rather than sideways) such that it hits the fingerboard.

\relative {
  c’4\snappizzicato
  <c’ e g>4\snappizzicato
  <c’ e g>4~\snappizzicato
  <c, e g>4_\snappizzicato
}

2.4 Fretted string instruments
Chapter 2: Specialist notation

2.4.1 Common notation for fretted strings

This section discusses common notation that is unique to fretted string instruments.

References for fretted strings

Music for fretted string instruments is normally notated on a single staff, either in traditional music notation or in tablature. Sometimes the two types are combined, and it is especially common in popular music to use chord diagrams above a staff of traditional notation. The guitar and the banjo are transposing instruments, sounding an octave lower than written. Scores for these instruments should use the "treble_8" clef (or \transposition c to get correct MIDI output). Some other elements pertinent to fretted string instruments are covered elsewhere:

- Fingerings are indicated as shown in [Fingering instructions], page 229.
- Instructions for Laissez vibrer ties as well as ties on arpeggios and tremolos can be found in [Ties], page 55.
- Instructions for handling multiple voices can be found in [Collision resolution], page 178.
- Instructions for indicating harmonics can be found in [Harmonics], page 352.

See also

Notation Reference: [Fingering instructions], page 229, [Ties], page 55, [Collision resolution], page 178, [Instrument names], page 210, [Writing music in parallel], page 188, [Arpeggio], page 145, Section A.14 [List of articulations], page 768, [Clef], page 17, [Instrument transpositions], page 27.

String number indications

The string on which a note should be played may be indicated by appending \number to a note.

\clef "treble_8"
c4\5 e\4 g2\3
<c\5 e\4 g3>1

When fingerings and string indications are used together, their placement can be controlled by the order in which the two items appear in the code only if they appear inside of an explicit
chord: applied to whole chords or single notes outside of chords, fingerings are placed using a different mechanism.

\clef "treble_8"
g4\3-0
g-\0\3
<g\3-0>
<g-0\3>

String numbers may also, as is customary with unfretted strings, be printed in Roman numerals and placed below the staff rather than above.

\clef "treble_8"
c'2\2
a\3
\romanStringNumbers
c'\2
\set stringNumberOrientations = #'(down)
a\3
\arabicStringNumbers
g1\4

Most behaviors of string number indications (namely, the StringNumber object), including their placement, may be set in the same way as fingerings: see [Fingering instructions], page 229.

Predefined commands
\arabicStringNumbers, \romanStringNumbers.

See also
Notation Reference: [Fingering instructions], page 229.
Snippets: Section “Fretted strings” in Snippets.
Internals Reference: Section “StringNumber” in Internals Reference, Section “Fingering” in Internals Reference.

Default tablatures
Music for plucked string instruments is frequently notated using a finger/touch notation or tablature. In contrast to traditional notation pitches are not denoted with note heads, but by numbers (or letter-like symbols in historical intavolatura). The staff lines in tablature indicate the string on which the note is to be played, and a number placed on a staff line indicated the fret at which the corresponding string is to be pressed. Notes that are to be played simultaneously are vertically aligned.

By default, string 1 is the highest string, and corresponds to the top line on the TabStaff. The tuning of the TabStaff strings defaults to the standard guitar tuning (with 6 strings).
The notes are printed as tablature, by using `TabStaff` and `TabVoice` contexts. A calligraphic tablature clef is added automatically.

\new TabStaff \relative {
  a,8 a' <c e> a
  d,8 a' <d f> a
}

Default tablatures do not contain any symbols for tone duration nor any other musical symbols such as expressive marks, for example.

\begin{Verbatim}
\texttt{\{\}
\}
\end{Verbatim}

If all musical symbols used in traditional notation should also show up in tablature one has to apply the command `\tabFullNotation` in a `TabStaff`-context. Please bear in mind that half notes are double-stemmed in tablature in order to distinguish them from quarter notes.

\begin{Verbatim}
\texttt{\{\}
\end{Verbatim}
By default pitches are assigned to the lowest playing position on the fret-board (first position). Open strings are automatically preferred. If you would like a certain pitch to be played on a specific string you can add a string number indication to the pitch name. If you don’t want to have string number indications appear in traditional notation, you can override the respective stencil. Usually it will be more comfortable to define the playing position by using the value of minimumFret. The default value for minimumFret is 0.

Even when minimumFret is set, open strings are used whenever possible. This behaviour can be changed by setting restrainOpenStrings to #t.

Chord constructs can be repeated by the chord repetition symbol q. In combination with tabulatures, its behavior of removing string and finger numbers alongside with other events is cumbersome, so you’ll want to run

\chordRepeats #'(string-number-event fingering-event)
explicitly on music expressions in tabulature using \texttt{Chord repetition}, page 170. This particular command is so common that it is available as \texttt{tabChordRepeats}.

\begin{verbatim}
guitar = \relative {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ 8 q4
}
\end{verbatim}

\begin{verbatim}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
\end{verbatim}

Ties over a line break are parenthesized by default. The same holds for the second alternative of a repeat.

\begin{verbatim}
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~
  }
  \alternative {
    \{ g4 f2. \}
    \{ g4\repeatTie c,2. \}
  }
  b1~
  \break
  b1
  \bar "."
}
\end{verbatim}

\begin{verbatim}
\score {
  <<
    \new StaffGroup <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
}
\end{verbatim}
The command \hideSplitTiedTabNotes cancels the behavior of engraving fret numbers in parentheses:

\scriptsize
\begin{verbatim}
ties = \relative { \\repeat volta 2 { 
  e'2. f4~
  2 g2~ }
\\alternative { 
  \{ g4 f2. \} 
  \{ g4\repeatTie c,2. \}
}
  b1~
  \break
  b1
  \bar "|."
}

\score { 
  << 
    \new StaffGroup <<
    \new Staff { 
      \clef "treble_8"
      \ties
    }
    \new TabStaff { 
      \hideSplitTiedTabNotes 
      \ties
    }
  }
>>
\end{verbatim}
\end{scriptsize}
Harmonic indications can be added to tablature notation as sounding pitches:

```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```

Note that the command \harmonic must always be attached to single notes (possibly inside of a chord) instead of whole chords. It only makes sense for open-string harmonics in the 12th
Chapter 2: Specialist notation

fret. All other harmonics should be calculated by LilyPond. This can be achieved by indicating the fret where a finger of the fretting hand should touch a string.

```lilypond
fretHarmonics = {\
    \harmonicByFret #5 d16\4\
    \harmonicByFret #4 d16\4\
    \harmonicByFret #3 d8\4\
    \harmonicByFret #5 <g\3 b\2>2.\
}\
\score {<<\
    \new Staff {\clef "treble_8"\
        \fretHarmonics\
    }\
    \new TabStaff { \fretHarmonics }\
    >>}
```

Alternatively, harmonics can be computed by defining the ratio of string lengths above and below the harmonic fingering.

```lilypond
ratioHarmonics = {\
    \harmonicByRatio #1/2 <g\3 b\2 e'\1>4\
    \harmonicByRatio #1/3 <g\3 b\2 e'\1>4\
    \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }\
}\
\score {<<\
    \new Staff {\clef "treble_8"\
        \ratioHarmonics\
    }\
    \new TabStaff { \ratioHarmonics }\
    >>}
```
Selected Snippets

**Stem and beam behavior in tablature**

The direction of stems is controlled the same way in tablature as in traditional notation. Beams can be made horizontal, as shown in this example.

\begin{verbatim}
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
g,,16 b d g b d g b
  }
}
\end{verbatim}

**Polyphony in tablature**

Polyphony is created the same way in a TabStaff as in a regular staff.

\begin{verbatim}
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  \new StaffGroup = "tab with traditional"
  \new Staff = "guitar traditional"
  \clef "treble_8"
  \context Voice = "upper" \upper
  \context Voice = "lower" \lower
}\end{verbatim}
Open string harmonics in tablature

This snippet demonstrates open-string harmonics

```plaintext
openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = #3
  \override TextSpanner.dash-fraction = #0.3
  \override TextSpanner.dash-period = #1

  % first harmonic
  \harmonicByFret #12 e,2 \6 \startTextSpan
  \harmonicByRatio #1/2 e,2 \6 \stopTextSpan

  % second harmonic
  \harmonicByFret #7 e,2 \6 \startTextSpan
  \harmonicByRatio #1/3 e,2 \6
  \harmonicByFret #19 e,2 \6
  \harmonicByRatio #2/3 e,2 \6 \stopTextSpan
  \harmonicByFret #19 < e,2 \5 a,\5 d\4 >
  \harmonicByRatio #2/3 < e,2 \5 a,\5 d\4 >

  % third harmonic
  \harmonicByFret #5 e,2 \6 \startTextSpan
  \harmonicByRatio #1/4 e,2 \6
  \harmonicByFret #24 e,2 \6
  \harmonicByRatio #3/4 e,2 \6 \stopTextSpan

  % fourth harmonic
  \harmonicByFret #4 e,2 \6 \startTextSpan
```

\harmonicByRatio #1/5 e,\6
\harmonicByFret #9 e,\6
\harmonicByRatio #2/5 e,\6
\harmonicByFret #16 e,\6
\harmonicByRatio #3/5 e,\6\stopTextSpan

\%fifth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "5th harm."
\harmonicByFret #3 e,\6\startTextSpan
\harmonicByRatio #1/6 e,\6\stopTextSpan
\break

\%sixth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "6th harm."
\harmonicByFret #2.7 e,\6\startTextSpan
\harmonicByRatio #1/7 e,\6\stopTextSpan

\%seventh harmonic
\override TextSpanner.bound-details.left.text = \markup\small "7th harm."
\harmonicByFret #2.3 e,\6\startTextSpan
\harmonicByRatio #1/8 e,\6\stopTextSpan

\%eighth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "8th harm."
\harmonicByFret #2 e,\6\startTextSpan
\harmonicByRatio #1/9 e,\6\stopTextSpan

\score {
<<
\new Staff
\with { \omit StringNumber } {
\new Voice {
\clef "treble_8"
\openStringHarmonics
}
}
\new TabStaff {
\new TabVoice {
\openStringHarmonics
}
}
>>
}

\timesig.C44
\clefs.tab
\clefs.G

1st harm. 2nd harm. 3rd harm.
Fretted-string harmonics in tablature

Demonstrates fretted-string harmonics in tablature

pinchedHarmonics = {
  \textSpannerDown
  \override TextSpanner.bound-details.left.text =
    \markup \halign \#-0.5 \tenny "PH"
  \override TextSpanner.style =
    #'dashed-line
  \override TextSpanner.dash-period = #0.6
  \override TextSpanner.bound-details.right.attach-dir = #1
  \override TextSpanner.bound-details.right.text =
    \draw-line #'(0 1)
  \override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4\markup \tenny "AH 16"
  <\parenthesize g g'\harmonic>4\markup \tenny "AH 17"
  <\parenthesize d' d''\harmonic>2\markup \tenny "AH 19"
  %pinched harmonics (PH)
  \pinchedHarmonics
  a'\harmonic>2\startTextSpan
  d''\harmonic>4
  e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g'4 g'\harmonic>4\markup \tenny "TH 17"
  <\parenthesize a'4 a'\harmonic>4\markup \tenny "TH 19"
  <\parenthesize c'3 c''\harmonic>2\markup \tenny "TH 17"
  %touch harmonics (TCH)
  a4( <e'\harmonic>2. )\markup \tenny "TCH"
}

frettedStrings = {
  %artificial harmonics (AH)
\harmonicByFret #4 g4\3
\harmonicByFret #5 d4\4
\harmonicByFret #7 g2\3
%pinched harmonics (PH)
\harmonicByFret #7 d2\4
\harmonicByFret #5 d4\4
\harmonicByFret #7 a4\5
%tapped harmonics (TH)
\harmonicByFret #5 d4\4
\harmonicByFret #7 d4\4
\harmonicByFret #5 g2\3
%touch harmonics (TCH)
a4 \harmonicByFret #9 g2\3

\score {
<<
\new Staff
\with { \omit StringNumber } {
\new Voice {
 \clef "treble_8"
 \harmonics
}
}
\new TabStaff {
 \new TabVoice {
 \frettedStrings
 }
}
>>
}

Slides in tablature
Slides can be typeset in both Staff and TabStaff contexts:

slides = {
c'8\3(\glissando d'8\3)
c'8\3\glissando d'8\3
\hideNotes
\grace { g16\glissando }
\unHideNotes
c'4\3
\afterGrace d'4\3\glissando {
\stemDown \hideNotes

Chapter 2: Specialist notation

Chord glissando in tablature

Slides for chords are indicated by default in both Staff and TabStaff. String numbers are necessary for TabStaff because automatic string calculations are different for chords and for single notes.

\score {
  \new Staff { 
    \clef "treble_8" \slides 
  }
  \new TabStaff { \slides }
}

\override Glissando.minimum-length = #4
\override Glissando.springs-and-rods = #ly:spanner::set-spacing-rods
\override Glissando.thickness = #2
\omit StringNumber
% or:
%\override StringNumber.stencil = ##f
}

\myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
}

\score {
  \new Staff {
    \clef "treble_8"
    \myMusic
  }
  \new TabStaff \myMusic
}

\score {
  \new Staff {

Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

\new TabStaff {
    \relative c' {
        d4( e\2)
        a( g)
    }
}

Hammer on and pull off using voices

The arc of hammer-on and pull-off is upwards in voices one and three and downwards in voices two and four:

\new TabStaff {
    \relative c' {
        << { \voiceOne g2( a) }
        \ \ { \voiceTwo a,( b) }
        >> \oneVoice
    }
}

\clef "treble_8"
\myMusic
\new TabStaff \with { \override Glissando.style = #'none } { \myMusic
}
**Hammer on and pull off using chords**

When using hammer-on or pull-off with chorded notes, only a single arc is drawn. However “double arcs” are possible by setting the `doubleSlurs` property to `#t`.

```latex
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    \<g' b>8( <a c> <g b>)
  }
}
```

![Diagram of hammer on and pull off using chords]

See also

Notation Reference: [Chord repetition], page 170, [Glissando], page 140, [Harmonics], page 352, [Stems], page 234, [Written-out repeats], page 161.

Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “TabNoteHead” in Internals Reference, Section “TabStaff” in Internals Reference, Section “TabVoice” in Internals Reference, Section “Beam” in Internals Reference.

**Known issues and warnings**

Chords are not handled in a special way, and hence the automatic string selector may easily select the same string for two notes in a chord.

In order to handle `\partCombine`, a `TabStaff` must use specially-created voices:

```latex
melodia = \partCombine { e4 g g g } { e4 e e e }
<<
\new TabStaff <<
  \new TabVoice = "one" s1
  \new TabVoice = "two" s1
  \new TabVoice = "shared" s1
  \new TabVoice = "solo" s1
  \{ \melodia \}
>>
```

![Diagram of multiple voices in a TabStaff]

Guitar special effects are limited to harmonics and slides.
Custom tablatures

LilyPond tablature automatically calculates the fret for a note based on the string to which the note is assigned. In order to do this, the tuning of the strings must be specified. The tuning of the strings is given in the `stringTunings` property.

LilyPond comes with predefined string tunings for banjo, mandolin, guitar, bass guitar, ukulele, violin, viola, cello, and double bass. LilyPond automatically sets the correct transposition for predefined tunings. The following example is for bass guitar, which sounds an octave lower than written.

\begin{code}
<<
\new Voice \with {
  \omit StringNumber
} {
  \clef "bass_8"
  \relative {
    c,4 d e f
  }
}
\new TabStaff \with {
  stringTunings = #bass-tuning
} {
  \relative {
    c,4 d e f
  }
}
>>
\end{code}

The default string tuning is `guitar-tuning`, which is the standard EAD-GBE tuning. Some other predefined tunings are `guitar-open-g-tuning`, `mandolin-tuning` and `banjo-open-g-tuning`. The predefined string tunings are found in `ly/string-tunings-init.ly`.

Any desired string tuning can be created. The \texttt{\stringTuning} function can be used to define a string tuning which can be used to set `stringTunings` for the current context.

Its argument is a chord construct defining the pitches of each string in the tuning. The chord construct must be in absolute octave mode, see [Absolute octave entry], page 1. The string with the highest number (generally the lowest string) must come first in the chord. For example, we can define a string tuning for a four-string instrument with pitches of a'', d'', g', and c':

\begin{code}
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b' c'''
}
\end{code}

\begin{code}
<<
\new Staff {
\end{code}
The `stringTunings` property is also used by `FretBoards` to calculate automatic fret diagrams.

String tunings are used as part of the hash key for predefined fret diagrams (see [Predefined fret diagrams], page 383).

The previous example could also be written as follows:

```latex
\begin{verbatim}
custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }

  \new TabStaff {
    \set TabStaff.stringTunings = #custom-tuning
    \mynotes
  }
  \mynotes
\end{verbatim}
```

Internally, a string tuning is a Scheme list of string pitches, one for each string, ordered by string number from 1 to N, where string 1 is at the top of the tablature staff and string N is at the bottom. This ordinarily results in ordering from highest pitch to lowest pitch, but some instruments (e.g., ukulele) do not have strings ordered by pitch.
A string pitch in a string tuning list is a LilyPond pitch object. Pitch objects are created with the Scheme function `ly:make-pitch` (see Section A.22 [Scheme functions], page 824).

The `\stringTuning` creates such an object from chord input.

LilyPond automatically calculates the number of lines in the TabStaff and the number of strings in an automatically calculated FretBoard as the number of elements in `stringTunings`.

To let all TabStaff contexts use the same custom tuning by default, you can use

```
\layout {
  \context {\TabStaff
   \stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

A modern tab clef can also be used.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```

```
\begin{tabular}{c|c}
\hline
T & A \\
\hline
2 & 2 \\
\hline
\end{tabular}
```

The modern tab clef supports tablatures from 4 to 7 strings.

TabStaff may support microtones like quarter-tones, which can be played using bendings. `supportNonIntegerFret = ##t` needs to be set in Score-context. However, microtones are not supported in FretBoards.

```
\layout {
  \context {\Score
   \supportNonIntegerFret = ##t
  }
}
```

```
\begin{tabular}{c|c}
\hline
T & A \\
\hline
2 & 2 \\
\hline
\end{tabular}
```

```
mus = \relative {
  ees4es'
eseh
  ees
  eeh
  e
  eih
```
See also

Notation Reference: [Absolute octave entry], page 1, [Predefined fret diagrams], page 383, Section A.22 [Scheme functions], page 824.

Installed Files: ly/string-tunings-init.ly, scm/tablature.scm.

Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “Tab_note_heads_ engraver” in Internals Reference.

Known issues and warnings

Automatic tablature calculations do not work properly in most cases for instruments where string pitches do not vary monotonically with string number, such as ukuleles.

Fret diagram markups

Fret diagrams can be added to music as a markup to the desired note. The markup contains information about the desired fret diagram. There are three different fret diagram markup interfaces: standard, terse, and verbose. The three interfaces produce equivalent markups, but have varying amounts of information in the markup string. Details about the syntax of the different markup strings used to define fret diagrams are found at Section A.11.5 [Instrument Specific Markup], page 750.

The standard fret diagram markup string indicates the string number and the fret number for each dot to be placed on the string. In addition, open and unplayed (muted) strings can be indicated.
Barre indications can be added to the diagram from the fret diagram markup string.

```
<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1^\markup {
    \fret-diagram "c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
  }
  <g, d g b d' g'>1^\markup {
    \fret-diagram "c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
  }
}
>>
```

The size of the fret diagram, and the number of frets in the diagram can be changed in the fret diagram markup string.

```
<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1^\markup {
    \fret-diagram "s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
  }
}
```
The number of strings in a fret diagram can be changed to accommodate different instruments such as banjos and ukuleles with the fret diagram markup string.

Fingering indications can be added, and the location of fingering labels can be controlled by the fret diagram markup string.
\fret-diagram "f:2;6-x;5-x;4-o;3-2;2-3-3;1-1-1;"

Dot radius and dot position can be controlled with the fret diagram markup string.

\new ChordNames {
\chordmode {
  c1 d:m
}
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1\markup {
    \fret-diagram "d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
  }
  <d a d' f'>1\markup {
    \fret-diagram "p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
  }
}

Fret-diagrams may be printed left-handed
\markup
\center-column {
  "C"
  "(left-handed)"
  \override #`(fret-diagram-details . ((handedness . ,LEFT)))
  \fret-diagram "6-x;5-3-3;4-2-2;3-o;2-1;1-o;"
The fret-diagram-terse markup string omits string numbers; the string number is implied by the presence of semicolons. There is one semicolon for each string in the diagram. The first semicolon corresponds to the highest string number and the last semicolon corresponds to the first string. Mute strings, open strings, and fret numbers can be indicated.

\new ChordNames { \chordmode { c1 d:m } } \new Staff { \clef "treble_8" <c e g c' e'>1 \markup { \fret-diagram-terse "x;3;2;0;1;0;" } <d a d' f'>1 \markup { \fret-diagram-terse "x;3;2;0;1;0;" } } }

Barre indicators can be included in the fret-diagram-terse markup string.

\new ChordNames { \chordmode { f1 g } } \new Staff { \clef "treble_8" <f, c f a c' f'>1 \markup { \fret-diagram-terse "1-(;3;3;2;1;1-);" } <g, d g b d' g'>1 \markup { \fret-diagram-terse "3-(;5;5;4;3;3-);" } }
Fingering indications can be included in the fret-diagram-terse markup string.

```latex
<< \new ChordNames { 
  \chordmode { 
    c1 d:m 
  } }
\new Staff { 
  \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string 
  \clef "treble_8" 
  <c e g c' e'>1^\markup { 
    \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" 
  }
  <d a d' f'>1^\markup { 
    \fret-diagram-terse "x;x;o;2-2;3-3;1-1;" 
  }
} >>
```

Other fret diagram properties must be adjusted using `\override` when using the fret-diagram-terse markup.

Only one indication per string can be included in a fret-diagram-terse markup. To have multiple indications per string use a fret diagram or fret-diagram-verbose markup.

The fret-diagram-verbose markup string is in the format of a Scheme list. Each element of the list indicates an item to be placed on the fret diagram.

```latex
<< \new ChordNames { 
  \chordmode { 
    c1 d:m 
  } }
\new Staff { 
  \clef "treble_8" 
  <c e g c' e'>1^\markup { 
    \fret-diagram-verbose #'( 
      (mute 6) 
      (place-fret 5 3) 
      (place-fret 4 2) 
      (open 3) 
      (place-fret 2 1) 
      (open 1) 
    ) 
  } 
} >>
```
Fingering indications and barres can be included in a fret-diagram-verbose markup string. Unique to the fret-diagram-verbose interface is a capo indication that can be placed on the fret diagram. The capo indication is a thick bar that covers all strings. The fret with the capo will be the lowest fret in the fret diagram.

Fingering indication dots can be colored as well as parenthesized; the parenthesis’s color can also be altered independently.

Markups can be placed into the dots as well.

<<
\new ChordNames {
  \chordmode {
    f1 g c c b
  }
}
\new Staff {
  \clef "treble_8"
  \override Voice.TextScript
    .fret-diagram-details.finger-code = #'below-string
  <f, c f a c' f'>1\markup {
    \fret-diagram-verbose #'(  
      (place-fret 6 1)  
      (place-fret 5 3)  
      (place-fret 4 3)  
      (place-fret 3 2)  
      (place-fret 2 1)  
      (place-fret 1 1)  
      (barre 6 1 1)
    )
  }
  <g, b, d g b g'>1\markup {
    \fret-diagram-verbose #'(  
      (place-fret 6 3 2)
    )
  }
>>
(place-fret 5 2 1)
(open 4)
(open 3)
(open 2)
(place-fret 1 3 3)
)

\fret-diagram-verbose \\
(capo 3)
(mute 6)
(place-fret 4 5 1)
(place-fret 3 5 2)
(place-fret 2 5 3)
)
\override Voice.TextScript.size = 1.4
<c g c' e' g'>1\markup {
\fret-diagram-verbose #'(
(place-fret 6 3 1 red parenthesized default-paren-color)
(place-fret 5 3 1 inverted)
(place-fret 4 5 2 blue parenthesized)
(place-fret 3 5 3 blue)
(place-fret 2 5 4 blue)
(place-fret 1 3 1 inverted)
)
}
\override Voice.TextScript.size = 1.5
<b, fis b dis' fis'>1\markup 
\fret-diagram-details . ((finger-code . in-dot)))
\fret-diagram-verbose \\
(place-fret 5 2 1)
(place-fret 4 4 "fis" red)
(place-fret 3 4 "b" red)
(place-fret 2 4
\concat {
\vcenter "d"
\fontsize #-5
\musicglyph "accidentals.sharp"}
red)
(place-fret 1 2 1)
)
}
All other fret diagram properties must be adjusted using \override when using the fret-diagram-verbose markup.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the fret-diagram-interface. Details are found at Section “fret-diagram-interface” in Internals Reference. For a fret diagram markup, the interface properties belong to Voice.TextScript.

Selected Snippets

*Changing fret orientations*

Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

\include "predefined-guitar-fretboards.ly"

<<
\chords {
  c1
  c
  c1
}
\new FretBoards {
  \chordmode {
    c1
    \override FretBoard.fret-diagram-details.orientation = #'landscape
    c1
    \override FretBoard.fret-diagram-details.orientation = #'opposing-landscape
    c1
  }
}
\new Voice {
  c'1
  c'1
  c'
}
>>
Customizing markup fret diagrams

Fret diagram properties can be set through 'fret-diagram-details. For markup fret diagrams, overrides can be applied to the Voice.TextScript object or directly to the markup.

\chords { c1 | c | c | d }
\new Voice = "mel" { 
\textLengthOn
\set global properties of fret diagram
\override TextScript.size = #'1.2
\override TextScript.fret-diagram-details.finger-code = #'in-dot
\override TextScript.fret-diagram-details.dot-color = #'white

%%%% C major for guitar, no barre, using defaults
% terse style
c'1\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1,o;" } 

%%%% C major for guitar, barred on third fret
% verbose style
% size 1.0
% roman fret label, finger labels below string, straight barre
c'1\markup {
% standard size
\override #'(size . 1.0) {
\override #'(fret-diagram-details . (  
  (number-type . roman-lower)  
  (finger-code . in-dot)  
  (barre-type . straight)) ) {
\fret-diagram-verbose #'(mute 6)  
  (place-fret 5 3 1)  
  (place-fret 4 5 2)  
  (place-fret 3 5 3)  
  (place-fret 2 5 4)  
  (place-fret 1 3 1)  
  (barre 5 1 3))
}
}
}

%%%% C major for guitar, barred on third fret
% verbose style
% landscape orientation, arabic numbers, M for mute string
% no barre, fret label down or left, small mute label font
c'1\markup {
\override #'(fret-diagram-details . (  
  (finger-code . below-string)  
  (number-type . arabic)  
  (label-dir . -1)  
  (mute-string . "M")  
  (orientation . landscape)  
  (barre-type . none)
See also

Notation Reference: Section A.11.5 [Instrument Specific Markup], page 750.

Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “fret-diagram-interface” in Internals Reference.

Predefined fret diagrams

Fret diagrams can be displayed using the FretBoards context. By default, the FretBoards context will display fret diagrams that are stored in a lookup table:

\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
  }
}
Chapter 2: Specialist notation

The default predefined fret diagrams are contained in the file `predefined-guitar-fretboards.ly`. Fret diagrams are stored based on the pitches of a chord and the value of `stringTunings` that is currently in use. `predefined-guitar-fretboards.ly` contains predefined fret diagrams only for `guitar-tuning`. Predefined fret diagrams can be added for other instruments or other tunings by following the examples found in `predefined-guitar-fretboards.ly`.

Fret diagrams for the ukulele are contained in the file `predefined-ukulele-fretboards.ly`.

\include "predefined-ukulele-fretboards.ly"

\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}

\include "predefined-mandolin-fretboards.ly"

\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}

Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], page 424).

\include "predefined-guitar-fretboards.ly"
It is common that both chord names and fret diagrams are displayed together. This is achieved by putting a ChordNames context in parallel with a FretBoards context and giving both contexts the same music.

Predefined fret diagrams are transposable, as long as a diagram for the transposed chord is stored in the fret diagram table.
The predefined fret diagram table for guitar contains eight chords (major, minor, augmented, diminished, dominant seventh, major seventh, minor seventh, dominant ninth) for each of 17 keys. The predefined fret diagram table for ukulele contains these chords plus an additional three chords (major sixth, suspended second, and suspended fourth). A complete list of the predefined fret diagrams is shown in Section A.4 [Predefined fretboard diagrams], page 671. If there is no entry in the table for a chord, the FretBoards engraver will calculate a fret diagram using the automatic fret diagram functionality described in [Automatic fret diagrams], page 393.

\include{predefined-guitar-fretboards.ly}
mychords = \chordmode{
c1 c:maj9
}

<<
\new ChordNames{
  mychords
}
\new FretBoards{
  mychords
}
>>

Fret diagrams can be added to the fret diagram table. To add a diagram, you must specify the hash table for the diagram, the chord for the diagram, the tuning to be used, and a definition for the diagram. Normally, the hash table will be default-fret-table. The diagram definition can be either a fret-diagram-terse definition string or a fret-diagram-verbose marking list.

\include{predefined-guitar-fretboards.ly}
\storePredefinedDiagram #default-fret-table
\chordmode{c:maj9}
#guitar-tuning
"x;3-2;0;0;0;0;"

mychords = \chordmode{
c1 c:maj9
}

<<
\new ChordNames{
  mychords
}
\new FretBoards{
  mychords
}
>>
Different fret diagrams for the same chord name can be stored using different octaves of pitches. The different octave should be at least two octaves above or below the default octave, because the octaves above and below the default octave are used for transposing fretboards.

\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
   \chordmode { c'' }
   #guitar-tuning
   #(offset-fret 2 (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
   c1 c''
}
<<
   \new ChordNames {
      \mychords
   }
   \new FretBoards {
      \mychords
   }
>>

In addition to fret diagrams, LilyPond stores an internal list of chord shapes. The chord shapes are fret diagrams that can be shifted along the neck to different positions to provide different chords. Chord shapes can be added to the internal list and then used to define predefined fret diagrams. Because they can be moved to various positions on the neck, chord shapes will normally not contain any open strings. Like fret diagrams, chord shapes can be entered as either fret-diagram-terse strings or fret-diagram-verbose marking lists.

\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

\addChordShape #'powerf #guitar-tuning "1-1;3-3;3-4;x;x;x;"

% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
   \chordmode { f'' }
   #guitar-tuning
   #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
   \chordmode { g'' }
   #guitar-tuning
The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Section “fret-diagram-interface” in Internals Reference. For a predefined fret diagram, the interface properties belong to `FretBoards.FretBoard`.

**Selected Snippets**

*Customizing fretboard fret diagrams*

Fret diagram properties can be set through `fret-diagram-details`. For FretBoard fret diagrams, overrides are applied to the `FretBoards.FretBoard` object. Like `Voice`, `FretBoards` is a bottom level context, therefore can be omitted in property overrides.

```latex
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
#guitar-tuning
"x;1-1-;3-2,3-3,3-4;1-1-;"

<<
\new ChordNames {
  \chordmode { c1 | c | c | d }
}
\new FretBoards {
  % Set global properties of fret diagram
  \override FretBoards.FretBoard.size = #'1.2
  \override FretBoard.fret-diagram-details.finger-code = #'in-dot
  \override FretBoard.fret-diagram-details.dot-color = #'white
  \chordmode { c
    \once \override FretBoard.size = #'1.0
    \once \override FretBoard.fret-diagram-details.barre-type = #'straight
    \once \override FretBoard.fret-diagram-details.dot-color = #'black
    \once \override FretBoard.fret-diagram-details.finger-code = #'below-string
    c'
    \once \override FretBoard.fret-diagram-details.barre-type = #'none
    \once \override FretBoard.fret-diagram-details.number-type = #'arabic
```
Defining predefined fretboards for other instruments

Predefined fret diagrams can be added for new instruments in addition to the standards used for guitar. This file shows how this is done by defining a new string-tuning and a few predefined fretboards for the Venezuelan cuatro.

This file also shows how fingerings can be included in the chords used as reference points for the chord lookup, and displayed in the fret diagram and the TabStaff, but not the music.

These fretboards are not transposable because they contain string information. This is planned to be corrected in the future.

% add FretBoards for the Cuatro
% Note: This section could be put into a separate file
% predefined-cuatro-fretboards.ly
% and \included into each of your compositions

\new Voice {
    c'1 | c' | c' | d'
}

\setFretBoardTuning { #`(,(ly:make-pitch 0 6 0)
    ,(ly:make-pitch 1 3 SHARP)
    ,(ly:make-pitch 1 1 0)
    ,(ly:make-pitch 0 5 0))
}

dSix = { <a\4 d\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 cis\1 e\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
\storePredefinedDiagram #default-fret-table \dSix
\storePredefinedDiagram #default-fret-table cuatrotuning
\storePredefinedDiagram #default-fret-table "o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
  #cuatroTuning
  \"o;o;o;3-3;\"
\storePredefinedDiagram #default-fret-table \aMajSeven
  #cuatroTuning
  \"o;2-2;1-1;2-3;\"
\storePredefinedDiagram #default-fret-table \dMajSeven
  #cuatroTuning
  \"o;o;o;1-1;\"
\storePredefinedDiagram #default-fret-table \gMajor
  #cuatroTuning
  \"2-2;o;1-1;o;\"

% end of potential include file /predefined-cuatro-fretboards.ly

%(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      primerosNames
    }

    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c' {
        primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
      \override FretBoard
      \override FretBoard.fret-diagram-details.string-count = 4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      primeros
    }

    \new TabStaff \relative c' {

ChordChanges for FretBoards

FretBoards can be set to display only when the chord changes or at the beginning of a new line.

\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

<< \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
Alternate fretboard tables can be created. These would be used in order to have alternate fretboards for a given chord.

In order to use an alternate fretboard table, the table must first be created. Fretboards are then added to the table.

The created fretboard table can be blank, or it can be copied from an existing table.

The table to be used in displaying predefined fretboards is selected by the property `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"

% Make a blank new fretboard table
#(define custom-fretboard-table-one (make-fretboard-table))

% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two (make-fretboard-table default-fret-table))

% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
  \chordmode {c}
  #guitar-tuning
  "3-(;3;5;5;5;3-);"

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
  \chordmode {c}
  #guitar-tuning
  "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
\new FretBoards {
  \chordmode {
    \set predefinedDiagramTable = #default-fret-table
    c1 | d1 |
    \set predefinedDiagramTable = #custom-fretboard-table-one
```
See also

Notation Reference: [Custom tablatures], page 370, [Automatic fret diagrams], page 393, [Chord mode overview], page 424, Section A.4 [Predefined fretboard diagrams], page 671.


Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “fret-diagram-interface” in Internals Reference.

Automatic fret diagrams

Fret diagrams can be automatically created from entered notes using the FretBoards context. If no predefined diagram is available for the entered notes in the active stringTunings, this context calculates strings and frets that can be used to play the notes.

\[\new \text{ChordNames} \{
\chordmode \{
\f1 \g
\}\}
\]
As no predefined diagrams are loaded by default, automatic calculation of fret diagrams is the default behavior. Once default diagrams are loaded, automatic calculation can be enabled and disabled with predefined commands:

\storePredefinedDiagram #default-fret-table
\chordmode {c1 c c}
\predefinedFretboardsOff
\chordmode {c e g c' e'}
\guitar-tuning
"x;3-1-;5-2;5-3;5-4;3-1-1-;"
Sometimes the fretboard calculator will be unable to find an acceptable diagram. This can often be remedied by manually assigning a note to a string. In many cases, only one note need be manually placed on a string; the rest of the notes will then be placed appropriately by the FretBoards context.

Fingerings can be added to FretBoard fret diagrams.

```
<< \new ChordNames { \chordmode { c1 d:m } } \new FretBoards { <c-3 e-2 g c'-1 e'>1 <d a-2 d'-3 f'-1>1 } \new Staff { \clef "treble_8" <c e g c' e'>1 <d a d' f'>1 } >>
```

The minimum fret to be used in calculating strings and frets for the FretBoard context can be set with the `minimumFret` property.

```
<< \new ChordNames { \chordmode { d1:m d:m } } \new FretBoards { <d a d' f'>1 \set FretBoards.minimumFret = #5 <d a d' f'>1 } \new Staff { \clef "treble_8" <d a d' f'>1 <d a d' f'>1 }
```
The strings and frets for the \texttt{FretBoards} context depend on the \texttt{stringTunings} property, which has the same meaning as in the \texttt{TabStaff} context. See [Custom tablatures], page 370, for information on the \texttt{stringTunings} property.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the \texttt{fret-diagram-interface}. Details are found at Section “fret-diagram-interface” in \textit{Internals Reference}. For a \texttt{FretBoards} fret diagram, the interface properties belong to \texttt{FretBoards.FretBoard}.

\textbf{Predefined commands}

\texttt{\predefinedFretboardsOff}, \texttt{\predefinedFretboardsOn}.

\textbf{See also}

Notation Reference: [Custom tablatures], page 370.

\textit{Snippets}: Section “Fretted strings” in \textit{Snippets}.

\textit{Internals Reference}: Section “fret-diagram-interface” in \textit{Internals Reference}.

\textbf{Known issues and warnings}

Automatic fretboard calculations do not work properly for instruments with non-monotonic tunings.

\textbf{Right-hand fingerings}

Right-hand fingerings \texttt{p-i-m-a} must be entered using \texttt{\rightHandFinger} followed by a number.

\begin{quote}
\textbf{Note:} If the number is entered in Scheme notation, remember to append a space before following it with a closing > or similar.
\end{quote}

\begin{verbatim}
clef "treble_8"
c\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c'\rightHandFinger #4
<\rightHandFinger #1 e\rightHandFinger #2
g\rightHandFinger #3 c'\rightHandFinger #4 >1
\end{verbatim}

For convenience, \texttt{\rightHandFinger} may be abbreviated to something shorter, for example \texttt{\RH}, by adding the appropriate definition at the source file’s top level:

\texttt{\RH = \rightHandFinger \etc}
Most behaviors of right-hand fingerings (namely, the StrokeFinger object) may be set in the same way as ordinary fingerings: see [Fingering instructions], page 229.

Selected Snippets

Placement of right-hand fingerings

It is possible to exercise greater control over the placement of right-hand fingerings by setting a specific property, as demonstrated in the following example.

```asciimusic
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  \set strokeFingerOrientations = #'(up down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4
  \set strokeFingerOrientations = #'(up right down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4
  \set strokeFingerOrientations = #'(left)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >2
  \set strokeFingerOrientations = #'(right)
  c\RH #1
}
```

Fingerings string indications and right-hand fingerings

This example combines left-hand fingering, string indications, and right-hand fingering.

```asciimusic
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5\RH #1 >4
  <e-2\4\RH #2 >4
  <g-0\3\RH #3 >4
  <c-1\2\RH #4 >4
}
```
See also

Notation Reference: [Fingering instructions], page 229.
Snippets: Section “Fretted strings” in Snippets.
Internals Reference: Section “StrokeFinger” in Internals Reference.

2.4.2 Guitar
Most of the notational issues associated with guitar music are covered sufficiently in the general fretted strings section, but there are a few more worth covering here. Occasionally users want to create songbook-type documents having only lyrics with chord indications above them. Since LilyPond is a music typesetter, it is not recommended for documents that have no music notation in them. A better alternative is a word processor, text editor, or, for experienced users, a typesetter like GuitarTeX.

Indicating position and barring
This example demonstrates how to include guitar position and barring indications.

```
\relative {
  \clef "treble_8"
  b,16 d g b e
  \textSpannerDown
  \override TextSpanner.bound-details.left.text = "XII"
  g16\startTextSpan
  b16 e g e b g\stopTextSpan
  e16 b g d
}
```

See also

Notation Reference: [Text spanners], page 246.
Snippets: Section “Fretted strings” in Snippets, Section “Expressive marks” in Snippets.

Indicating harmonics and dampened notes
Special note heads can be used to indicate dampened notes or harmonics. Harmonics are normally further explained with a text markup.

```
\relative {
  \clef "treble_8"
  \override NoteHead.style = #'harmonic-mixed
d'8}\markup { \italic \fontsize #-2 "harm. 12" } <g b>4
}
```

```
\relative {
  \clef "treble_8"
  \override NoteHead.style = #'harmonic-mixed
d'8}\markup { \italic \fontsize #-2 "harm. 12" } <g b>4
}
```
Dampened notes (also called *dead notes*) are supported within normal and tablature staves:

```lilypond
music = \relative { 
  < a\3 \deadNote c\2 a'\1 >4 
  < b\3 \deadNote d\2 b'\1 > 
  < c\3 \deadNote e\2 c'\1 > 
  \deadNotesOn 
  \tuplet 3/2 { g8 b e } 
  \deadNotesOff 
  < a,\3 c\2 e\1 >1 
} 
\new StaffGroup << 
\new Staff { 
  \clef "treble_8"
  \music 
} 
\new TabStaff { 
  \music 
} 
>> 
```

Another playing technique (especially used on electric guitars) is called *palm mute*. The string is hereby partly muted by the palm of the striking hand (hence the name). LilyPond supports the notation of palm mute-style notes by changing the note head to a triangle shape.

```lilypond
\new Voice { % Warning: explicit Voice instantiation is 
% required to have palmMuteOff work properly 
% when palmMuteOn comes at the beginning of 
% the piece. 
\relative c, { 
  \clef "G_8"
  \palmMuteOn 
  e8\markup { \musicglyph "noteheads.u2do" = palm mute } 
  < e b' e > e 
  \palmMuteOff 
  e e \palmMute e e e | 
  e8 \palmMute { e e e } e e e e | 
  < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2 
} 
} 
```
Indicating power chords

Power chords and their symbols can be engraved in chord mode or as chord constructs. As an exception, the fifth is specified in these chord names, whereas it is usually left out in other chords (e.g. major or minor triads).

```latex
ChordsAndSymbols = {  
    \chordmode {  
        e,,1:5  
        a,,:5.8  
        \set TabStaff.restrainOpenStrings = ##t  
        \set minimumFret = #8  
        c,,:5  
        f,:5.8  
    }  
    \set minimumFret = #2  
    \set restrainOpenStrings = ##f  
    <a, e> <a cis' e'>  
    <g d' g'>  
}  
\score {  
    <<  
        \new ChordNames {  
            \ChordsAndSymbols  
        }  
        \new Staff {  
            \clef "treble_8"  
            \ChordsAndSymbols  
        }  
        \new TabStaff {  
            \ChordsAndSymbols  
        }  
    >>  
}  
```

See also

Snippets: Section “Fretted strings” in Snippets.

Notation Reference: [Special note heads], page 39, Section A.9 [Note head styles], page 705.
See also

Music Glossary: Section “power chord” in Music Glossary.
Notation Reference: [Extended and altered chords], page 426, [Printing chord names], page 429.
Snippets: Section “Fretted strings” in Snippets.

2.4.3 Banjo

Banjo tablatures

LilyPond has basic support for the five-string banjo. When making tablatures for five-string banjo, use the banjo tablature format function to get correct fret numbers for the fifth string:

```lilypond
music = {
  g8 d' g'\5 a b g e d' |
  g4 d'\8 b' a'\5 g'\5 e'\2 d' |
  g4
}
<<
\new Staff \with { \omit StringNumber }
{ \clef "treble_8" \music }
\new TabStaff \with {
  tablatureFormat = #fret-number-tablature-format-banjo
  stringTunings = #banjo-open-g-tuning
}
{ \music }
>>
```

A number of common tunings for the five-string banjo are predefined: \texttt{banjo-c-tuning} (gCGBD), \texttt{banjo-modal-tuning} (gDGCD), \texttt{banjo-open-d-tuning} (aDF#AD) and \texttt{banjo-open-dm-tuning} (aDFAD).

These may be converted to four-string tunings using the \texttt{four-string-banjo} function:

\texttt{\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)}

See also

Installed Files: \texttt{ly/string-tunings-init.ly}.
Snippets: Section “Fretted strings” in Snippets.

2.4.4 Lute

Lute tablatures

LilyPond supports tablature for lute.

To get additional bass strings use \texttt{additionalBassStrings}, where the pitches of those strings are set. They will be printed below lowest line as: a, /a, //a, ///a, 4, 5, etc.
fret-letter-tablature-format for tablatureFormat should be used, probably fretLabels for further customizing.

\[ m = \{ f'4 d' a f d a, g, fis, e, d, c, \bar "|." \} \]

\score { << \new Staff { \clef bass \cadenzaOn \m } \new TabStaff \m >> \layout { \context { \Score tablatureFormat = #fret-letter-tablature-format } \context { \TabStaff stringTunings = \stringTuning <a, d f a d' f'> additionalBassStrings = \stringTuning <c, d, e, fis, g,> fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k") } } }

Known issues and warnings

Using FretBoards with additionalBassStrings is not supported and will yield unsatisfying results.

2.5 Percussion

2.5.1 Common notation for percussion

Rhythmic music is primarily used for percussion and drum notation, but it can also be used to show the rhythms of melodies.

References for percussion

- Some percussion may be notated on a rhythmic staff; this is discussed in [Showing melody rhythms], page 82, and [Instantiating new staves], page 191.
- MIDI output is discussed in a separate section; please see Section 3.5 [Creating MIDI output], page 531.

See also

Notation Reference: [Showing melody rhythms], page 82, [Instantiating new staves], page 191. Section 3.5 [Creating MIDI output], page 531.
Snippets: Section “Percussion” in Snippets.

Basic percussion notation

Percussion notes may be entered in \drummode mode, which is similar to the standard mode for entering notes. The simplest way to enter percussion notes is to use the \drums command, which creates the correct context and entry mode for percussion:

```latex
\drums {
  hihat4 hh bassdrum bd
}
```

This is shorthand for:

```latex
\new DrumStaff \drummode {
  hihat4 hh bassdrum bd
}
```

Each piece of percussion has a full name and an abbreviated name, and both can be used in input files. The full list of percussion note names may be found in Section A.15 [Percussion notes], page 770.

Note that the normal notation of pitches (such as cis4) in a DrumStaff context will cause an error message. Percussion clefs are added automatically to a DrumStaff context but they can also be set explicitly. Other clefs may be used as well.

```latex
\drums {
  \clef percussion
  bd4 4 4 4
  \clef treble
  hh4 4 4 4
}
```

There are a few issues concerning MIDI support for percussion instruments; for details please see Section 3.5 [Creating MIDI output], page 531.

See also

Notation Reference: Section 3.5 [Creating MIDI output], page 531, Section A.15 [Percussion notes], page 770.

Installed Files: ly/drumpitch-init.ly.

Snippets: Section “Percussion” in Snippets.
**Drum rolls**

Drum rolls are indicated with three slashes across the stem. For quarter notes or longer the three slashes are shown explicitly, eighth notes are shown with two slashes (the beam being the third), and drum rolls shorter than eighths have one stem slash to supplement the beams. This is achieved with the tremolo notation, as described in [Tremolo repeats], page 166.

```
\drums {
  \time 2/4
  sn16 8 16 8 8:32 ~
  8 8 4:32 ~
  4 8 16 16
  4 r4
}
```

Sticking can be indicated by placing a markup for "R" or "L" above or below notes, as discussed in Section 5.4.2 [Direction and placement], page 634. The staff-padding property may be overridden to achieve a pleasing baseline.

```
\drums {
  \repeat unfold 2 {
    sn16"^"L" 16"^"R" 16"^"L" 16"^"R" 16"^"L" 16"^"R" 16"^"L" 16"^"R" 16"^"L" 16"^"R"
    \stemUp
    sn16\_\_"L" 16\_\_"R" 16\_\_"L" 16\_\_"R" 16\_\_"L" 16\_\_"R" 16\_\_"L" 16\_\_"R" 16\_\_"R"
  }
}
```

**See also**

Notation Reference: [Tremolo repeats], page 166.
Snippets: Section “Percussion” in *Snippets*.

**Pitched percussion**

Certain pitched percussion instruments (e.g., xylophone, vibraphone, and timpani) are written using normal staves. This is covered in other sections of the manual.

**See also**

Notation Reference: Section 3.5 [Creating MIDI output], page 531.
Snippets: Section “Percussion” in *Snippets*.

**Percussion staves**

A percussion part for more than one instrument typically uses a multiline staff where each position in the staff refers to one piece of percussion. To typeset the music, the notes must be interpreted in `DrumStaff` and `DrumVoice` context.

```
up = \drummode {
```
The above example shows verbose polyphonic notation. The short polyphonic notation, described in Section “I’m hearing Voices” in Learning Manual, can also be used. For example,

\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
  }
  \repeat unfold 16 hh16
  \drummode {
    bd4 sn4 bd4 sn4
  } >>

There are also other layout possibilities. To use these, set the property \texttt{drumStyleTable} in context \texttt{DrumVoice}. The following variables have been predefined:

\texttt{drums-style}

This is the default. It typesets a typical drum kit on a five-line staff:
The drum scheme supports six different toms. When there are fewer toms, simply select the toms that produce the desired result. For example, to get toms on the three middle lines you use `tommh`, `tomml`, and `tomfh`.

**agostini-drums-style**

Invented by the French percussionist Dante Agostini in 1965, this notation is commonly employed in France but also elsewhere.

**timbales-style**

This typesets timbales on a two line staff:

**congas-style**

This typesets congas on a two line staff:

**bongos-style**

This typesets bongos on a two line staff:
percussion-style

To typeset all kinds of simple percussion on one line staves:

\[ \text{tritriotrimguiquisguilcbcltambcabmarhc} \]

Custom percussion styles may also be defined, as explained in [Custom percussion staves], page 407.

See also

Notation Reference: [Custom percussion staves], page 407.
Installed Files: ly/drumpitch-init.ly.
Snippets: Section “Percussion” in Snippets.

Custom percussion staves

Custom percussion styles may be defined, to which the drumStyleTable property may then be set. Existing notations may be redefined as an association list where each entry has to be comprised of four items: a name, the note head style (or default), an articulation sign if needed (or #f if not), and the note head’s position on the staff. That list must then be converted into a Scheme hash table, using the alist->hash-table function.

```
#(define mydrums '(
  (bassdrum default #f -1)
  (snare default #f 0)
  (hihat cross #f 1)
  (halfopenhihat cross "halfopen" 1)
  (pedalhihat xcircle "stopped" 2)
  (lowtom diamond #f 3))
```

New names may also be added to these custom notations through the drumPitchNames variable, that may be redefined as an association list (or augmented by appending a new list to its existing value, as demonstrated below), but also through its individual entries. This also makes it possible to define aliases: alternate input shorthand for some notations.

```
drumPitchNames =
  #(append
    '(((leftsnap . sidestick)
```
In a similar manner, the `drumPitchTable` property associates a specific pitch (meaning a different instrument sound, as provided by available MIDI soundfonts) to each notation. That property needs to be defined as a hash table, which is again converted from an association list (stored by default as the `midiDrumPitches` variable). Redefining these associations is achieved as explained above, either by defining an entire association list or through individual entries. The following example demonstrates how to create a whole notation set with its own input syntax, custom notations and corresponding MIDI output.

```scheme
(define djembe-style
  '((dbass default #f -2)
    (dbassmute default "stopped" -2)
    (dopen default #f 0)
    (dopenmute default "stopped" 0)
    (dslap default #f 2)
    (dslapmute default "stopped" 2)))

midiDrumPitches.dbass = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap = b
midiDrumPitches.dslapmute = ais

test = \drummode { dba4 do ds dbm dom dsm }
```

```
\score {
```
See also

Installed Files: `ly/drumpitch-init.ly`.
Snippets: Section “Percussion” in `Snippets`.
Internals Reference: Section “DrumStaff” in `Internals Reference`, Section “DrumVoice” in `Internals Reference`.

**Ghost notes**

Also known as dead, muted, silenced or false notes; ghost notes can be created using the `\parenthesize` command, see [Parentheses], page 234.

\new DrumStaff <<
\new DrumVoice = "1" { s1 }
\new DrumVoice = "2" { s1 }
\drummode {
  <<
  { hh8[ 8] <hh sn> hh16 \parenthesize sn hh \parenthesize sn hh8 <hh sn> hh }
  { bd4 r4 bd8 8 r8 bd }
  } >>
} >>

See also

Notation Reference: [Parentheses], page 234.
Snippets: Section “Percussion” in `Snippets`.
2.6 Wind instruments

This section includes elements of music notation that arise when writing specifically for wind instruments.

2.6.1 Common notation for wind instruments

This section discusses notation common to most wind instruments.

References for wind instruments

Many notation issues for wind instruments pertain to breathing and tonguing:

- Breathing can be specified by rests or [Breath marks], page 138.
- Legato playing is indicated by [Slurs], page 133.
- Different types of tonguings, ranging from legato to non-legato to staccato are usually shown by articulation marks, sometimes combined with slurs, see [Articulations and ornamentations], page 122, and Section A.14 [List of articulations], page 768.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See [Tremolo repeats], page 166.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see [Instrument transpositions], page 27.
- Slide glissandi are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], page 140.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as [Grace notes], page 115.
- Pitch inflections at the end of a note are discussed in [Falls and doits], page 140.
- Key slaps or valve slaps are often shown by the cross style of [Special note heads], page 39.
- Woodwinds can overblow low notes to sound harmonics. These are shown by the flageolet articulation. See Section A.14 [List of articulations], page 768.
- The use of brass mutes is usually indicated by a text markup, but where there are many rapid changes it is better to use the stopped and open articulations. See [Articulations and ornamentations], page 122, and Section A.14 [List of articulations], page 768.
- Stopped horns are indicated by the stopped articulation. See [Articulations and ornamentations], page 122.
Selected Snippets

Changing \flageolet mark size

To make the \flageolet circle smaller use the following tweak.

\begin{verbatim}
smallFlageolet = \tweak font-size -3 \flageolet
\end{verbatim}

\layout { ragged-right = ##f }

\begin{verbatim}
\relative c' {
  d4^\flageolet \markup { default size } d_\flageolet
  c4^\smallFlageolet \markup { smaller } c_\smallFlageolet
}
\end{verbatim}

See also

Notation Reference: [Breath marks], page 138, [Slurs], page 133, [Articulations and ornamentations], page 122, Section A.14 [List of articulations], page 768, [Tremolo repeats], page 166, [Instrument transpositions], page 27, [Glissando], page 140, [Grace notes], page 115, [Falls and doits], page 140, [Special note heads], page 39,

Snippets: Section “Winds” in Snippets.

Fingerings

All wind instruments other than the trombone require the use of several fingers to produce each pitch. Some fingering examples are shown in the snippets below.

Woodwind diagrams can be produced and are described in Section 2.6.3.1 [Woodwind diagrams], page 415.

Selected Snippets

Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

\begin{verbatim}
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \relative c' {
    g\open
    \once \override TextScript.staff-padding = #-1.0
    \centermarkup
    g^\markup {
      \combine
        \musicglyph "scripts.open"
        \musicglyph "scripts.tenuto"
      }
}
\end{verbatim}
chapter 2: specialist notation

The following example demonstrates how fingering charts for wind instruments can be realized.

% range chart for paetzold contrabass recorder

centermarkup = {
\once \override TextScript.self-alignment-X = #CENTER
\once \override TextScript.X-offset = #(lambda (g)
  (+ (ly:self-alignment-interface::centered-on-x-parent g)
    (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
\new Staff \with {
  \remove "Time_signature_engraver"
  \omit Stem
  \omit Flag
  \consists "Horizontal_bracket_engraver"
}
{
  \clef bass
  \set Score.timing = ##f
  f,1*1/4 \glissando
  \clef violin
  gis'1*1/4
  \stemDown a'4\markup {1}
  \centermarkup \once \override TextScript.padding = #2
  bes'1*1/4 \markup \override #'(baseline-skip . 1.7) \column
    { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 3 \finger 4 \finger 5 \finger 6 \finger 7 }
  b'1*1/4
  c'4\markup {1}
  \centermarkup \once \override TextScript.padding = #2
  cis'1*1/4
}
See also

Notation Reference: Section 2.6.3.1 [Woodwind diagrams], page 415.
Snippets: Section “Winds” in Snippets.

2.6.2 Bagpipes

This section discusses notation common bagpipes.

Bagpipe definitions

LilyPond contains special definitions for Scottish, Highland Bagpipe music; to use them, add

```
\include "bagpipe.ly"
```

to the top of your input file. This lets you add the special grace notes common to bagpipe music with short commands. For example, you could write `\taor` instead of

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` also contains pitch definitions for the bagpipe notes in the appropriate octaves, so you do not need to worry about `\relative` or `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```

Bagpipe music nominally uses the key of D Major (even though that isn’t really true). However, since that is the only key that can be used, the key signature is normally not written out. To set this up correctly, always start your music with `\hideKeySignature`. If you for some reason want to show the key signature, you can use `\showKeySignature` instead.

Some modern music use cross fingering on c and f to flatten those notes. This can be indicated by `c-flat` or `f-flat`. Similarly, the piobaireachd high g can be written `g-flat` when it occurs in light music.
See also

Snippets: Section “Winds” in Snippets.

Bagpipe example

This is what the well known tune Amazing Grace looks like in bagpipe notation.

\include "bagpipe.ly"
layout {
  indent = 0.0\cm
  \context { \Score \remove "Bar_number_engraver" } }

\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{\hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrwd d2.
  \slurd d2
  \bar "|."}

Amazing Grace

Hymn

Trad. arr.
2.6.3 Woodwinds

This section discusses notation specifically for woodwind instruments.

2.6.3.1 Woodwind diagrams

Woodwind diagrams can be used to indicate the fingering to be used for specific notes and are available for the following instruments:

- piccolo
- flute
- oboe
- clarinet
- bass clarinet
- saxophone
- bassoon
- contrabassoon

Woodwind diagrams are created as markups:

```
c''1\markup {
    \woodwind-diagram #'piccolo #'((lh . (gis))
    (cc . (one three))
    (rh . (ees)))
}
```

Keys can be open, partially-covered, ring-depressed, or fully covered:

```
\textLengthOn
c''1\markup {
    \center-column {
        "one quarter"
        \woodwind-diagram #'flute #'((cc . (oneq))
        (lh . ()))
    }
```

See also

Snippets: Section “Winds” in Snippets.
Trills are indicated as shaded keys:
A variety of trills can be displayed:

\textLengthOn
\c""1\markup {\center-column {\text "one quarter to ring" \woodwind-diagram #'flute #((cc . (one1qTR)) (lh . () (rh . ()))}}}

\c""1\markup {\center-column {\text "ring to shut" \woodwind-diagram #'flute #((cc . (oneTR)) (lh . () (rh . ()))}}}

\c""1\markup {\center-column {\text "ring to open" \woodwind-diagram #'flute #((cc . (oneRT)) (lh . () (rh . ()))}}}

\c""1\markup {\center-column {\text "open to shut" \woodwind-diagram #'flute #((cc . (oneT)) (lh . () (rh . ()))}}
The list of all possible keys and settings for a given instrument can be displayed on the console using

```lilypond
#(print-keys-verbose 'flute)
```

or in the log file using

```lilypond
#(print-keys-verbose 'flute (current-error-port))
```

although they will not show up in the music output.

Creating new diagrams is possible, although this will require Scheme ability and may not be accessible to all users. The patterns for the diagrams are in `scm/define-woodwind-diagrams.scm` and `scm/display-woodwind-diagrams.scm`.

**Predefined commands**

**Selected Snippets**

**Woodwind diagrams listing**

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```lilypond
\layout {
  indent = 0
}

\relative c' {
  \textLengthOn
  c1``
Graphical and text woodwind diagrams

In many cases, the keys other than the central column can be displayed by key name as well as by graphical means.

\relative c' {
\textLengthOn
\relative c' {c1}{\override #'(graphical . #f) {
  \woodwind-diagram
  #'piccolo
  '#'((cc . (one three))
    (lh . (gis))
    (rh . (ees)))
}\markup}
  \woodwind-diagram
  #'piccolo
  #'((cc . (one three))
    (lh . (gis))
    (rh . (ees)))
}\markup}
}
Changing the size of woodwind diagrams

The size and thickness of woodwind diagrams can be changed.

\relative c' { 
\textLengthOn
\mark c1 \markup 
\woodwind-diagram
  #'piccolo
  #'()

\mark c2 \markup
\override #'(size . 1.5) {
  \woodwind-diagram
  #'piccolo
  #'()
}

\mark c3 \markup
\override #'(thickness . 0.15) {
  \woodwind-diagram
  #'piccolo
  #'()
}
}

Woodwind diagrams key lists

The snippet below produces a list of all possible keys and key settings for woodwind diagrams as defined in scm/define-woodwind-diagrams.scm. The list will be displayed in the log file, but not in the music. If output to the console is wanted, omit the \texttt{(current-error-port)} from the commands.

\texttt{(print-keys-verbose 'piccolo (current-error-port))}
Chapter 2: Specialist notation

#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))

\score {c''1}

See also

Installed Files: scm/define-woodwind-diagrams.scm,
scm/display-woodwind-diagrams.scm.

Snippets: Section “Winds” in Snippets.


2.7 Chord notation

\faffcf \baf f c \cacc f
1. Fair is the sun - shine, Fair - er the moon - light
2. Fair are the mead - ows, Fair - er the wood - land,

\fbafff \caccf \baf c
And all the stars in heav'n a bove;
Robed in the flow - ers of bloom - ing spring;

\flat \aff \caccf \baf \cacc

---

\flat \aff \caccf \baf \cacc

---

\flat \aff \caccf \baf \cacc

---

\flat \aff \caccf \baf \cacc

---
Chords can be entered either as normal notes or in chord mode and displayed using a variety of traditional European chord naming conventions. Chord names and figured bass notation can also be displayed.

### 2.7.1 Chord mode

Chord mode is used to enter chords using an indicator of the chord structure, rather than the chord pitches.

#### Chord mode overview

Chords can be entered as simultaneous music, as discussed in [Chorded notes], page 168.

Chords can also be entered in “chord mode”, which is an input mode that focuses on the structures of chords in traditional European music, rather than on specific pitches. This is convenient for those who are familiar with using chord names to describe chords. More information on different input modes can be found at Section 5.4.1 [Input modes], page 632.

```latex
\chordmode \{ c1 g a g c \}
```

Chords entered using chord mode are music elements, and can be transposed just like chords entered using simultaneous music. `\chordmode` is absolute, as `\relative` has no effect on `\chordmode` blocks. However, in `\chordmode` the absolute pitches are one octave higher than in note mode.

Chord mode and note mode can be mixed in sequential music:

```latex
\relative {
  <c' e g>2 <g b d>
  \chordmode \{ c2 f \}
  <c e g>2 <g' b d>
  \chordmode \{ f2 g \}
}
```

See also


Notation Reference: [Chorded notes], page 168, Section 5.4.1 [Input modes], page 632.

Snippets: Section “Chords” in Snippets.

### Known issues and warnings

Predefined shorthands for articulations and ornaments cannot be used on notes in chord mode, see [Articulations and ornamentations], page 122.
Common chords

Major triads are entered by including the root and an optional duration:

\chordmode { c2 f4 g }

Minor, augmented, and diminished triads are entered by placing : and a quality modifier string after the duration:

\chordmode { c2:m f4:aug g:dim }

Seventh chords can be created:

\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }

The table below shows the actions of the quality modifiers on triads and seventh chords. The default seventh step added to chords is a minor or flatted seventh, which makes the dominant seventh the basic seventh chord. All alterations are relative to the dominant seventh. A more complete table of modifier usage is found at Section A.2 [Common chord modifiers], page 667.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The default action; produces a major triad.</td>
<td><img src="image1" alt="Minor Triad" /></td>
</tr>
<tr>
<td>m, m7</td>
<td>The minor chord. This modifier lowers the 3rd.</td>
<td><img src="image2" alt="Minor Triad" /></td>
</tr>
<tr>
<td>dim, dim7</td>
<td>The diminished chord. This modifier lowers the 3rd, 5th and (if present) the 7th step.</td>
<td><img src="image3" alt="Diminished Triad" /></td>
</tr>
<tr>
<td>aug</td>
<td>The augmented chord. This modifier raises the 5th step.</td>
<td><img src="image4" alt="Augmented Triad" /></td>
</tr>
</tbody>
</table>
maj, maj7  The major 7th chord. This modifier adds a raised 7th step. The 7 following maj is optional. Do NOT use this modifier to create a major triad.

See also
Notation Reference: Section A.2 [Common chord modifiers], page 667, [Extended and altered chords], page 426.
Snippets: Section “Chords” in Snippets.

Known issues and warnings
Only one quality modifier should be used per chord, typically on the highest step present in the chord. Chords with more than quality modifier will be parsed without an error or warning, but the results are unpredictable. Chords that cannot be achieved with a single quality modifier should be altered by individual pitches, as described in [Extended and altered chords], page 426.

Extended and altered chords
Chord structures of arbitrary complexity can be created in chord mode. The modifier string can be used to extend a chord, add or remove chord steps, raise or lower chord steps, and add a bass note or create an inversion.

The first number following the : is taken to be the extent of the chord. The chord is constructed by sequentially adding thirds to the root until the specified number has been reached. Note that the seventh step added as part of an extended chord will be the minor or flatted seventh, not the major seventh. If the extent is not a third (e.g., 6), thirds are added up to the highest third below the extent, and then the step of the extent is added. The largest possible value for the extent is 13. Any larger value is interpreted as 13.

\chordmode {
  c1:2  c:3  c:4  c:5
  c1:6  c:7  c:8  c:9
  c1:10 c:11 c:12 c:13
  c1:14
}

As a special exception, c:5 produces a ‘power chord’ only consisting of root and fifth.

Since an unaltered 11 does not sound good when combined with an unaltered 13, the 11 is removed from a :13 chord (unless it is added explicitly).

\chordmode {
  c1:13  c:13.11  c:m13
}

As a special exception, c:5 produces a ‘power chord’ only consisting of root and fifth.
Individual steps can be added to a chord. Additions follow the extent and are prefixed by a dot (\textbullet{}). The basic seventh step added to a chord is the minor or flatted seventh, rather than the major seventh.

\begin{chordmode}
c1:3.5.6 c:3.7.8 c:3.6.13
\end{chordmode}

Added steps can be as high as desired.

\begin{chordmode}
c4:3.5.15 c:3.5.20 c:3.5.25 c:3.5.30
\end{chordmode}

Added chord steps can be altered by suffixing a \textit{-} or \textit{+} sign to the number. To alter a step that is automatically included as part of the basic chord structure, add it as an altered step.

\begin{chordmode}
c1:7+ c:5+.3- c:3-.5-.7-
\end{chordmode}

Following any steps to be added, a series of steps to be removed is introduced in a modifier string with a prefix of \textit{^}. If more than one step is to be removed, the steps to be removed are separated by \textit{^} following the initial \textit{^}.

\begin{chordmode}
c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
\end{chordmode}

The modifier \textit{sus} can be added to the modifier string to create suspended chords. This removes the 3rd step from the chord. Append either 2 or 4 to add the 2nd or 4th step to the chord. When \textit{sus} is followed by either a 2nd or 4th step, it is equivalent to \textit{^}3, otherwise to \textit{sus4}, namely 5.4.

\begin{chordmode}
\end{chordmode}
Added bass notes (putting a pitch other than the root on the bottom of the chord) can be specified by appending `/pitch` to the chord.

\chordmode { c'1 c'/e c'/f }

If the added pitch is already part of the chord, this may be used to print chord inversions, in which case the pitch is not added but merely moved to the bottom of the chord. It may however be treated as an added note (and thus printed twice), by using the syntax `/+pitch`.

\chordmode { c'1 c'/g c'/+e }

Automatic chord inversions and voicings are demonstrated in [Chord inversions and specific voicings], page 429.

Chord modifiers that can be used to produce a variety of standard chords are shown in Section A.2 [Common chord modifiers], page 667.

See also

Notation Reference: [Chord inversions and specific voicings], page 429, Section A.2 [Common chord modifiers], page 667.

Snippets: Section “Chords” in Snippets.

Known issues and warnings

Each step can only be present in a chord once. The following simply produces the augmented chord, since 5+ is interpreted last.

\chordmode { c1:3.5.5-.5+ }
Chord inversions and specific voicings

In addition to chord modifiers and added bass notes, various functions may be used to automatically print chords in a specific inversion or voicing – for example the so-called ‘drop 2’ voicing commonly used in jazz music.

\chordmode {
  \dropNote 2 {
    c2:maj7 d:m7
  }
  \invertChords 1 d1:maj7
}

Unlike added bass notes shown in [Extended and altered chords], page 426, this only affects the way chords are printed on a staff, and not chord names written with letters. Furthermore, these functions may be used not only in chord mode but also with <...> chords constructs explained in [Chorded notes], page 168.

See also

Notation Reference: [Extended and altered chords], page 426, [Chorded notes], page 168.
Snippets: Section “Chords” in Snippets.

2.7.2 Displaying chords

Chords can be displayed by name, in addition to the standard display as notes on a staff.

Printing chord names

Chord names are printed in the ChordNames context:

\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}

C F G

Chords can be entered as simultaneous notes or through the use of chord mode. The displayed chord name will be the same, regardless of the mode of entry, unless there are inversions or added bass notes:

chordmusic = \relative { 
  <c' e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
}

<<
\new ChordNames {
  \chordmusic
Rests passed to a ChordNames context will cause the noChordSymbol markup to be displayed.

\chords { ... } is a shortcut notation for \new ChordNames \chordmode { ... }.

\chords {
  c2 f4.:m g8:maj7
}

C   Fm G^\Delta

\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}

C   Fm G^\Delta
Selected Snippets

*Showing chords at changes*

By default, every chord entered is printed; this behavior can be modified so that chord names are printed only at the start of lines and when the chord changes.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}

<< \new ChordNames {
  \set chordChanges = ##t
  harmonies
}
\new Staff {
  \relative c' { \harmonies }
}
>>
```

![Cm chord](image)

![Chord changes](image)

*Simple lead sheet*

When put together, chord names, a melody, and lyrics form a lead sheet:

```
<< \chords { c2 g:sus4 f e }
\new Staff \relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free _ _ }
>>
```

![Lead sheet](image)

One day this shall be free_

See also

Notation Reference: [Writing music in parallel], page 188.
Snippets: Section “Chords” in Snippets.
Internals Reference: Section “ChordNames” in Internals Reference, Section “ChordName” in Internals Reference, Section “Chord_name_engraver” in Internals Reference, Section “Volta_engraver” in Internals Reference, Section “Bar_engraver” in Internals Reference.
Known issues and warnings

Chords containing inversions or altered bass notes are not named properly if entered using simultaneous music.

Customizing chord names

There is no unique system for naming chords. Different musical traditions use different names for the same set of chords. There are also different symbols displayed for a given chord name. The names and symbols displayed for chord names are customizable.

The basic chord name layout is a system for Jazz music, proposed by Klaus Ignatzek (see Section “Literature list” in Essay). (Other chord naming systems may be implemented through Scheme functions, as demonstrated by the “Chord names alternative” snippet in Section “Chords” in Snippets.) A list of common jazz chords notations may be found on the chart in Section A.1 [Chord name chart], page 666.

The default naming system may be tweaked easily in a number of ways. To begin with, predefined commands allow to use different languages for the root pitch. These include \germanChords, \semiGermanChords, \italianChords and \frenchChords:

\begin{verbatim}
default E/D Cm B/B B\#B\# B/b/B/b
german E/d Cm H/h H\#his B/b
semi-german E/d Cm H/h H\#his B/b
italian Mi/Re Do m Si/Si Si\#/Si\# Si/b/Si/b
french Mi/Ré Do m Si/Si Si\#/Si\# Si/b/Si/b
\end{verbatim}

German songbooks may indicate minor chords as lowercase letters, without any m suffix. This can be obtained by setting the chordNameLowercaseMinor property:

\begin{verbatim}
\set chordNameLowercaseMinor = ##t
c2 d:m e:m f
}
\end{verbatim}

\begin{verbatim}
C d e F
\end{verbatim}

The chord name display can also be tuned through the following properties.

chordRootNamer

The chord name is usually printed as a letter for the root with an optional alteration. The transformation from pitch to letter is done by this function. Special note names (for example, the German ‘H’ for a B-chord) can be produced by storing a new function in this property.

majorSevenSymbol

This property contains the markup object used to follow the output of chordRootNamer to identify a major 7 chord. Predefined options are whiteTriangleMarkup and blackTriangleMarkup.
**additionalPitchPrefix**

When the chord name contains additional pitches, they can optionally be prefixed with some text. The default is no prefix, in order to avoid too much visual clutter, but for small numbers of additional pitches this can be visually effective.

\new ChordNames {
\<c e g d'>< % add9
\set additionalPitchPrefix = "add"
\<c e g d'>< % add9
}

\Cadd9

**chordNoteNamer**

When the chord name contains additional pitches other than the root (e.g., an added bass note), this function is used to print the additional pitch. By default the pitch is printed using chordRootNamer. The chordNoteNamer property can be set to a specialized function to change this behavior. For example, the bass note can be printed in lower case.

**chordNameSeparator**

Different parts of a chord name are normally separated by a small amount of horizontal space. By setting chordNameSeparator, you can use any desired markup for a separator. This does not affect the separator between a chord and its bass note; to customize that, use slashChordSeparator.

\chords {
\c4:7.9- \c:7.9-/g
\set chordNameSeparator = \markup { \\/ }
\break
\c4:7.9- \c:7.9-/g
}

\C7\b9  \C7\b9/G
\C7\b9  \C7\b9/G

**slashChordSeparator**

Chords can be played over a bass note other than the conventional root of the chord. These are known as “inversions” or “slash chords”, because the default way of notating them is with a forward slash between the main chord and the bass note. Therefore the value of slashChordSeparator defaults to a forward slash, but you can change it to any markup you choose.

\chords {
\c4:7.9- \c:7.9-/g
\set slashChordSeparator = \markup { " over " }
\break
\c4:7.9- \c:7.9-/g
}

\C7\b9  \C7\b9/G
\C7\b9  \C7\b9 over G
chordNameExceptions
This property is a list of pairs. The first item in each pair is a set of pitches used to identify the steps present in the chord. The second item is a markup that will follow the chordRootNamer output to create the chord name.

minorChordModifier
Minor chords are often denoted via a ‘m’ suffix to the right of the root of the chord. However some idioms prefer other suffices, such as a minus sign.

\chords {
  c4:min f:min7
  \set minorChordModifier = \markup { "-" }
  \break
  c4:min f:min7
}

Cm Fm7
C- F-

chordPrefixSpacer
The modifier for minor chords as determined by minorChordModifier is usually printed immediately to the right of the root of the chord. A spacer can be placed between the root and the modifier by setting chordPrefixSpacer. The spacer is not used when the root is altered.

Predefined commands
\whiteTriangleMarkup, \blackTriangleMarkup, \germanChords, \semiGermanChords, \italianChords, \frenchChords.

Selected Snippets
Chord name exceptions
The property chordNameExceptions can be used to store a list of special notations for specific chords.

% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
g1:maj9 g1:6.9
}
\layout {
   ragged-right = ##t
}

<< \context ChordNames \theMusic
\context Voice \theMusic
>>

\chordmode {
   f1:maj7 f:7 bes:7
}

F\textsuperscript{\Delta} | F\textsuperscript{7} | B\textsubscript{b}\textsuperscript{7} |

Chord name major7

The layout of the major 7 can be tuned with majorSevenSymbol.

\chords {
   c:7+
   \set majorSevenSymbol = \markup { j7 }
   c:7+
}

C\textsuperscript{\Delta} C\textsuperscript{7}

Adding bar lines to ChordNames context

To add bar line indications in the ChordNames context, add the Bar_engraver.

\score {
   \chords {
      c1
      c1
   }
   \new Staff \with {
      \consists "Volta_engraver"
   }
   \repeat volta 2 { c'1 }
   \alternative { c' }
}

Volta below chords

By adding the Volta_engraver to the relevant staff, volte can be put under chords.
Changing chord separator

The separator between different parts of a chord name can be set to any markup.
\chords {
  c:7sus4
  \set chordNameSeparator = \markup { \typewriter | }
  c:7sus4
}
C7sus4 C7|sus4

See also

Notation Reference: Section A.1 [Chord name chart], page 666, Section A.2 [Common chord modifiers], page 667.

Essay on automated music engraving: Section “Literature list” in Essay.


Snippets: Section “Chords” in Snippets.

Known issues and warnings

Chord names are determined from both the pitches that are present in the chord and the information on the chord structure that may have been entered in \chordmode. If the simultaneous pitches method of entering chords is used, undesired names result from inversions or bass notes.

myChords = \relative c' {
  \chordmode { ci c/g c/f }
  \<c e g>1 \<g c e> \<f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
}
2.7.3 Figured bass

LilyPond has support for figured bass, also called thorough bass or basso continuo:

```
\new Voice { \clef bass dis4 c d ais g fis}
\new FiguredBass {
  \figuremode {
    < 6 >4 < 7\+ >8 < 6+ [\_!] >
    < 6 >4 <6 5 [3+] >
    < \_ >4 < 6 5/>4
  }
```

Introduction to figured bass

Figured bass notation can be displayed.
Chapter 2: Specialist notation

The support for figured bass consists of two parts: there is an input mode, introduced by \texttt{\figuremode}, that accepts entry of bass figures, and there is a context named \texttt{FiguredBass} that takes care of displaying \texttt{BassFigure} objects. Figured bass can also be displayed in \texttt{Staff} contexts.

\texttt{\figures \{ ... \}} is a shortcut notation for \texttt{\new FiguredBass \figuremode \{ ... \}}.

Although the support for figured bass may superficially resemble chord support, it is much simpler. \texttt{\figuremode} mode simply stores the figures and the \texttt{FiguredBass} context prints them as entered. There is no conversion to pitches.

See also

Music Glossary: Section “figured bass” in \textit{Music Glossary}.

Snippets: Section “Chords” in \textit{Snippets}.

Entering figured bass

\texttt{\figuremode} is used to switch the input mode to figure mode. More information on different input modes can be found at Section 5.4.1 [Input modes], page 632.

In figure mode, a group of bass figures is delimited by \texttt{<} and \texttt{>}. The duration is entered after the \texttt{>}.

\texttt{\new FiguredBass \{}\texttt{\figuremode \{}\texttt{<6 4>2}}\texttt{\}}\texttt{\}\texttt{}\texttt{\}}\texttt{\texttt{\}}

\texttt{6 4}

Accidentals (including naturals) may be used for modifying scale steps. These are entered by appending + (for sharps), - (for flats) or ! (for naturals) after the number. For double accidentals the modifier is applied twice. For the modification of the third step the number is often omitted, which can be achieved by using _ instead of a number.

\texttt{\figures \{\texttt{<7! 6+ 4-> <5++> <3--> <_+ > < 7 _!>}}\texttt{\}}\texttt{\}

\texttt{\#7 \#5 \#3 \#7\#6 \#1}

Augmented and diminished steps can be indicated:

\texttt{\figures \{\texttt{<6\+ 5/> <7/>}}\texttt{\}}\texttt{\texttt{\}}
A backward slash through a figure (typically used for raised sixth steps) can be created:
\begin{verbatim}
\figures {
  <6> <6\>
}
\end{verbatim}

Brackets can be included in figures:
\begin{verbatim}
\figures {
  <[12] 8 [6 4]>
}
\end{verbatim}

Any text markup can be inserted as a figure:
\begin{verbatim}
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
\end{verbatim}

Continuation lines can be used to indicate repeated figures:
\begin{verbatim}
<<<
\{  
  \clef bass  
  e4 d c b, e4 d c b, 
\} \figures { 
  \bassFigureExtendersOn 
  <6 4>4 <6 3> <7 3> <7 3> 
  \bassFigureExtendersOff 
  <6 4>4 <6 3> <7 3> <7 3> 
}\end{verbatim}

In this case, the extender lines replace existing figures, unless the continuation lines have been explicitly terminated.
\begin{verbatim}
<<<
\figures {  
  \bassFigureExtendersOn 
  <6 4>4 <6 4> <6\! 4\!> <6 4>
}\end{verbatim}
The table below summarizes the figure modifiers available.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, !</td>
<td>Accidentals</td>
<td>( \frac{7}{5} \times 5 \times 3 )</td>
</tr>
<tr>
<td>*</td>
<td>Augmented and diminished steps</td>
<td>+6 7</td>
</tr>
<tr>
<td>\   \</td>
<td>Raised sixth step</td>
<td>6</td>
</tr>
<tr>
<td>!</td>
<td>End of continuation line</td>
<td>( \frac{6}{4} \times 4 \times 4 )</td>
</tr>
</tbody>
</table>

Predefined commands
\bassFigureExtendersOn, \bassFigureExtendersOff.

Selected Snippets

Changing the positions of figured bass alterations

Accidentals and plus signs can appear before or after the numbers, depending on the figuredBassAlterationDirection and figuredBassPlusDirection properties.

\begin{verbatim}
\figures {
  <6\+> <5++> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5++> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5++> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5++> <6 4-> r
}
\end{verbatim}
See also

Snippets: Section “Chords” in Snippets.

Internals Reference: Section “BassFigure” in Internals Reference, Section “BassFigureAlignment” in Internals Reference, Section “BassFigureLine” in Internals Reference, Section “BassFigureBracket” in Internals Reference, Section “BassFigureContinuation” in Internals Reference, Section “FiguredBass” in Internals Reference.

Displaying figured bass

Figured bass can be displayed using the \texttt{FiguredBass} context, or in most staff contexts.

When displayed in a \texttt{FiguredBass} context, the vertical location of the figures is independent of the notes on the staff.

```latex
<< \relative { c'4 c'8 r8 c4 c'} \new FiguredBass { \figuremode { <4>4 <10 6>8 s8 <6 4>4 <6 4> } } >>
```

In the example above, the \texttt{FiguredBass} context must be explicitly instantiated to avoid creating a second (empty) staff.

Figured bass can also be added to \texttt{Staff} contexts directly. In this case, the vertical position of the figures is adjusted automatically.

```latex
<< \new Staff = "myStaff" \figuremode { <4>4 <10 6>8 s8 <6 4>4 <6 4> } \context Staff = "myStaff" { \clef bass \c4 c'8 r8 c4 c' } >>
```

In the example above, the \texttt{FiguredBass} context must be explicitly instantiated to avoid creating a second (empty) staff.
When added in a Staff context, figured bass can be displayed above or below the staff.

\new Staff = "myStaff"
\figuremode {
  <4>4 <10 6>8 s8
  \bassFigureStaffAlignmentDown
  <6 4>4 <6 4>
}
%% Put notes on same Staff as figures
\context Staff = "myStaff" {
  \clef bass
  c4 c'8 r8 c4 c'
}

Predefined commands

\bassFigureStaffAlignmentDown, \bassFigureStaffAlignmentUp, \bassFigureStaffAlignmentNeutral.

See also

Snippets: Section “Chords” in Snippets.

Internals Reference: Section “BassFigure” in Internals Reference, Section “BassFigureAlignment” in Internals Reference, Section “BassFigureLine” in Internals Reference, Section “BassFigureBracket” in Internals Reference, Section “BassFigureContinuation” in Internals Reference, Section “FiguredBass” in Internals Reference.

Known issues and warnings

To ensure that continuation lines work properly, it is safest to use the same rhythm in the figure line as in the bass line.

\{
  \clef bass
  \repeat unfold 4 { f16. g32 } f8. es16 d8 es
\}
\figures {
  \bassFigureExtendersOn
  \repeat unfold 4 { <6 4->16. <6 4->32 }
  <5>8. r16 <6>8 <6\! 5->
}

\}
Chapter 2: Specialist notation

2.8 Contemporary music

From the beginning of the 20th Century there has been a massive expansion of compositional style and technique. New harmonic and rhythmic developments, an expansion of the pitch spectrum and the development of a wide range of new instrumental techniques have been accompanied by a parallel evolution and expansion of musical notation. The purpose of this section is to provide references and information relevant to working with these new notational techniques.

2.8.1 Pitch and harmony in contemporary music

This section highlights issues that are relevant to notating pitch and harmony in contemporary music.

References for pitch and harmony in contemporary music

- Standard quarter-tone notation is addressed in [Note names in other languages], page 8.
- Non-standard key signatures are addressed in [Key signature], page 22.
- Contemporary practises in displaying accidentals are addressed in [Automatic accidentals], page 29.

Microtonal notation

Contemporary key signatures and harmony

2.8.2 Contemporary approaches to rhythm

This section highlights issues that are relevant to the notation of rhythm in contemporary music.

References for contemporary approaches to rhythm

- Compound time signatures are addressed in [Time signature], page 66.
- Basic polymetric notation is addressed in [Polymetric notation], page 77.
- Feathered beams are addressed in [Feathered beams], page 98.
Menstruch notes (bar lines between staves only) are addressed in [Grouping staves], page 193.

Tuplets in contemporary music

Contemporary time signatures

Extended polymetric notation

Beams in contemporary music

Bar lines in contemporary music

2.8.3 Graphical notation

Rhythmic items may be continued by a duration line, which gets represented by a `DurationLine` grob. Possible styles are `beam`, `line`, `dashed-line`, `dotted-line`, `zigzag`, `trill` and `none`. The duration line may end with a hook (beam-style only) or an arrow.

```latex
\layout {
  \context {
    \Voice
    \consists "Duration_line_engraver"
    \omit Stem
    \omit Flag
    \omit Beam
    \override NoteHead.duration-log = 2
  }
}

\once \override DurationLine.style = #'line
a'1\- s2 r
\once \override DurationLine.style = #'dashed-line
a'1\- s2 r
\once \override DurationLine.style = #'dotted-line
a'1\- s2 r
\once \override DurationLine.style = #'zigzag
a'1\- s2 r
\once \override DurationLine.style = #'trill
a'1\- s2 r
\once \override DurationLine.style = #'none
a'1\- s2 r
\once \override DurationLine.bound-details.right.end-style = #'arrow
a'1\- s2 r
\override DurationLine.bound-details.right.end-style = #'hook
a'1\- s2 r
\override DurationLine.details.hook-direction = #DOWN
a'1\- s2 r
```
\bar "|."

\once \override DurationLine.bound-details.right.end-style = #'arrow
\lastEndStyle #'arrow
a'1\-\b\bar "|."

2.8.4 Contemporary scoring techniques
2.8.5 New instrumental techniques

2.8.6 Further reading and scores of interest

This section suggests books, musical examples and other resources useful in studying contemporary musical notation.

Books and articles on contemporary musical notation

- *Instrumentation and Orchestration* by Alfred Blatter [Schirmer, 2nd ed. 1997]

Scores and musical examples

2.9 Ancient notation

Salve, Regina, mater misericordiae: Ad te clamamus, exules, filii Hevae. Ad te suspiramus, gementes et flentes in hac lacrimarum valle. Eia ergo, Advocata nostra, illos tuos misericordes oculos ad nos converte.

Et Jesus, benedic tum fructum ventris tui, nobis post hoc exsilium ostende. O clemens: O
Support for ancient notation includes features for mensural notation, Gregorian chant notation, and Kievan square notation. These features can be accessed either by modifying style properties of graphical objects such as note heads and rests, or by using one of the pre-defined contexts for these styles.

Many graphical objects, such as note heads and flags, accidentals, time signatures, and rests, provide a style property, which can be changed to emulate several different styles of ancient notation. See

- [Mensural note heads], page 453,
- [Mensural accidentals and key signatures], page 455,
- [Mensural rests], page 454,
- [Mensural clefs], page 451,
- [Gregorian clefs], page 458,
- [Mensural flags], page 454,
- [Mensural time signatures], page 452.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], page 449,
- [Divisiones], page 459,
- [Ligatures], page 448.

**See also**


Notation Reference: [Mensural note heads], page 453, [Mensural accidentals and key signatures], page 455, [Mensural rests], page 454, [Gregorian clefs], page 458, [Mensural flags], page 454, [Mensural time signatures], page 452, [Custodes], page 449, [Divisiones], page 459, [Ligatures], page 448.

### 2.9.1 Overview of the supported styles

Three styles are available for typesetting Gregorian chant:

- *Editio Vaticana* is a complete style for Gregorian chant, following the appearance of the Solesmes editions, the official chant books of the Vatican since 1904. LilyPond has support for all the notational signs used in this style, including ligatures, custodes, and special signs such as the quilisma and the oriscus.

- The *Editio Medicea* style offers certain features used in the Medicea (or Ratisbona) editions which were used prior to the Solesmes editions. The most significant differences from the Vaticana style are the clefs, which have downward-slanted strokes, and the note heads, which are square and regular.

- The *Hufnagel* (“horseshoe nail”) or Gothic style mimics the writing style in chant manuscripts from Germany and Central Europe during the middle ages. It is named after the basic note shape (the virga), which looks like a small nail.

Three styles emulate the appearance of late-medieval and renaissance manuscripts and prints of mensural music:

- The *Mensural* style most closely resembles the writing style used in late-medieval and early renaissance manuscripts, with its small and narrow, diamond-shaped note heads and its rests which approach a hand-drawn style.
• The Neomensural style is a modernized and stylized version of the former: the note heads are broader and the rests are made up of straight lines. This style is particularly suited, e.g., for incipits of transcribed pieces of mensural music.

• The Petrucci style is named after Ottaviano Petrucci (1466-1539), the first printer to use movable type for music (in his Harmonice musices odhecaton, 1501). The style uses larger note heads than the other mensural styles.

Baroque and Classical are not complete styles but differ from the default style only in some details: certain note heads (Baroque) and the quarter rest (Classical).

Only the mensural style has alternatives for all aspects of the notation. Thus, there are no rests or flags in the Gregorian styles, since these signs are not used in plainchant notation, and the Petrucci style has no flags or accidentals of its own.

Each element of the notation can be changed independently of the others, so that one can use mensural flags, petrucci note heads, classical rests and vaticana clefs in the same piece, if one wishes.

See also


2.9.2 Ancient notation—common features

Pre-defined contexts

For Gregorian chant and mensural notation, there are pre-defined voice and staff contexts available, which set all the various notation signs to values suitable for these styles. If one is satisfied with these defaults, one can proceed directly with note entry without worrying about the details on how to customize a context. See one of the pre-defined contexts VaticanaVoice, VaticanaStaff, MensuralVoice, and MensuralStaff. See further

• [Gregorian chant contexts], page 457,
• [Mensural contexts], page 450.

See also


Notation Reference: [Gregorian chant contexts], page 457, [Mensural contexts], page 450.

Ligatures

A ligature is a graphical symbol that represents at least two distinct notes. Ligatures originally appeared in the manuscripts of Gregorian chant notation to denote ascending or descending sequences of notes on the same syllable. They are also used in mensural notation.

Ligatures are entered by enclosing them in \{ and \}. Some ligature styles may need additional input syntax specific for this particular type of ligature. By default, the LigatureBracket engraver just puts a square bracket above the ligature.

```
relative {
    \[ g' c, a' f d' \]
    a g f
    \[ e f a g \]
}
```

\[ e f a g \]
Two other ligature styles are available: the Vaticana for Gregorian chant, and the Mensural for mensural music (only white mensural ligatures are supported for mensural music, and with certain limitations). To use any of these styles, the default `Ligature_bracket_engraver` has to be replaced with one of the specialized ligature engravers in the `Voice` context, as explained in [White mensural ligatures], page 456, and [Gregorian square neume ligatures], page 461.

**See also**


Notation Reference: [White mensural ligatures], page 456, [Gregorian square neume ligatures], page 461.

**Known issues and warnings**

Spacing required for ligatures is not currently implemented and, as a result, there may end up being too much space between them. Line breaking may also be unsatisfactory.

Lyrics might not align as expected when using ligatures.

Accidentals must not be printed within a ligature, but instead be collected and printed in front of it.

The syntax still uses the deprecated “infix” style `\[ music expr \]`. For consistency reasons, it will eventually be changed to “postfix” style `note[ ... note]`.

**Custodes**

A *custos* (plural: *custodes*; Latin word for “guard”) is a symbol that appears at the end of a staff. It anticipates the pitch of the first note of the following line, thus helping the performer to manage line breaks during performance.

Custodes were frequently used in music notation until the seventeenth century. Nowadays, they have survived only in a few particular forms of musical notation such as contemporary editions of Gregorian chant like the *Editio Vaticana*. There are different custos glyphs used in different flavors of notational style.

For typesetting custodes, just put a `Custos_engraver` into the `Staff` context when declaring the `\layout` block, and change the style of the custos with an `\override` if desired, as shown in the following example:

```latex
\score {
  \relative {a'1}
  \break g
}
\layout {
  \context {\Staff \consists "Custos_engraver"
            \override Custos.style = #'mensural
          }
}  
```

![Example notation]
The custos glyph is selected by the `style` property. The styles supported are `vaticana`, `medicaea`, `hufnagel`, and `mensural`.

\new Lyrics \lyricmode {
\markup { \column {
  \typewriter "vaticana"
  \line { " " \musicglyph "custodes.vaticana.u0" }
} }
\markup { \column {
  \typewriter "medicaea"
  \line { " " \musicglyph "custodes.medicaea.u0" }
} }
\markup { \column {
  \typewriter "hufnagel"
  \line { " " \musicglyph "custodes.hufnagel.u0" }
} }
\markup { \column {
  \typewriter "mensural"
  \line { " " \musicglyph "custodes.mensural.u0" }
} }
}

\new Lyrics \lyricmode {
\markup { \column {
  \typewriter "vaticana"
  \line { " " \musicglyph "custodes.vaticana.u0" }
} }
\markup { \column {
  \typewriter "medicaea"
  \line { " " \musicglyph "custodes.medicaea.u0" }
} }
\markup { \column {
  \typewriter "hufnagel"
  \line { " " \musicglyph "custodes.hufnagel.u0" }
} }
\markup { \column {
  \typewriter "mensural"
  \line { " " \musicglyph "custodes.mensural.u0" }
} }
}

See also

Snippets: Section “Ancient notation” in Snippets.
Internals Reference: Section “Custos” in Internals Reference.

2.9.3 Typesetting mensural music

Mensural contexts

The predefined `MensuralVoice` and `MensuralStaff` contexts can be used to engrave a piece in mensural style. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

\score {
  \new MensuralVoice = "discantus" \relative {
    \hide Score.BarNumber {
      c'\'1\melisma \text{bes a g\melismaEnd}
      f\breve
      \f1\melisma \text{a c\breve d\melismaEnd}\]
      c\longa
      c\breve\melisma a1 g1\melismaEnd
      fis\longa\^\signumcongruentiae
    }
  }
}
\new Lyrics \lyricsto "discantus" {
    San -- ctus, San -- ctus, San -- ctus
}

\override NoteHead.style = #'vaticana.punctum
\clef "vaticana-do1" c'1

\override NoteHead.style = #'medicaea.punctum
\clef "medicaea-do3" c'1

\override NoteHead.style = #'hufnagel.punctum
\clef "hufnagel-fa2" c'1

\override NoteHead.style = #'neomensural
\clef "neomensural-c4" c'1

See also

Mensural clefs
Mensural clefs are supported using the \clef command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line.
It is possible to manually force a clef glyph to be typeset on an arbitrary line, as described in [Clef], page 17. For the complete range of possible clefs see Section A.10 [Clef styles], page 706.

**See also**


Notation Reference: [Gregorian clefs], page 458, [Clef], page 17.

Installed Files: `scm/parser-clef.scm`.

Snippets: Section “Pitches” in *Snippets*.


**Known issues and warnings**

The Mensural g clef is mapped to the Petrucci g clef.

**Mensural time signatures**

There is limited support for mensuration signs (which are similar to, but not exactly the same as time signatures). The glyphs are hard-wired to particular time fractions. In other words, to get a particular mensuration sign with the `\time n/m` command, `n` and `m` have to be chosen according to the following table:

```
\time 4/4 \time 2/2 \time 6/4 \time 6/8
\time 3/2 \time 3/4 \time 9/4 \time 9/8
\time 4/8 \time 2/4
```

Use the `style` property of grob `TimeSignature` to select ancient time signatures. Supported styles are `neomensural` and `mensural`. The above table uses the `neomensural` style. The following examples show the differences in style:

```
default numbered mensural neomensural
```

[Time signature], page 66, gives a general introduction to the use of time signatures.

**See also**


Notation Reference: [Time signature], page 66.
Known issues and warnings

Ratios of note durations cannot change with the time signature, as those are not constant. For example, the ratio of 1 breve = 3 semibreves (*tempus perfectum*) can be made by hand, by setting

```
breveTP = #(ly:make-duration -1 0 3/2)
...{ c\breveTP f1 }`
```

This sets `breveTP` to 3/2 times 2 = 3 times a whole note.

The `mensural68alt` and `neomensural68alt` symbols (alternate symbols for 6/8) are not addressable with \time. Use `\markup {\musicglyph "timesig.mensural68alt" }` instead.

Mensural note heads

For ancient notation, a note head style other than the default style may be chosen. This is accomplished by setting the *style* property of the NoteHead object to *baroque*, *neomensural*, *mensural*, *petrucci*, *blackpetrucci* or *semipetrucci*.

The *baroque* style differs from the default style by:

- Providing a maxima note head, and
- Using a square shape for \breve note heads.

The *neomensural*, *mensural*, and *petrucci* styles differ from the *baroque* style by:

- Using rhomboidal heads for semibreves and all smaller durations, and
- Centering the stems on the note heads.

The *blackpetrucci* style produces note heads usable in black mensural notation or coloratio sections in white mensural notation. Because note head style does not influence flag count, in this style a semiminima should be notated as \a8\*2, not \a4, otherwise it will look like a minima. The multiplier can be different if coloratio is used, e.g., to notate triplets.

Use *semipetrucci* style to draw half-colored note heads (breves, longas and maximas).

The following example demonstrates the *petrucci* style:

```
\compressEmptyMeasures
\autoBeamOff
\override NoteHead.style = #'petrucci
\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```

Section A.9 [Note head styles], page 705, gives an overview of all available note head styles.

See also


Notation Reference: Section A.9 [Note head styles], page 705.
Mensural flags

Use the \flag-style property of grob Stem to select ancient flags. Besides the default flag style, only the mensural style is supported.

\relative c' { 
  \override Flag.style = #'mensural
  \override Stem.thickness = #1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8 
  c'8 d e f c16 d e f c32 d e f
}

\relative c' {
  \override Flag.style = #'mensural
  \override Stem.thickness = #1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8 
  c'8 d e f c16 d e f c32 d e f
}

Note that the innermost flare of each mensural flag is vertically aligned with a staff line.

There is no particular flag style for neo-mensural or Petrucci notation. There are no flags in Gregorian chant notation.

See also


Known issues and warnings

Vertically aligning each flag with a staff line assumes that stems always end either exactly on or exactly in the middle of two staff lines. This may not always be true when using advanced layout features of classical notation (which however are typically out of scope for mensural notation).

Mensural rests

Use the style property of grob Rest to select ancient rests. Supported ancient styles are neomensural, and mensural.

The following example demonstrates these styles:

\compressEmptyMeasures 
\override Rest.style = #'mensural
r\longa""mensural" r\breve r1 r2 r4 r8 r16 s \break 
\override Rest.style = #'neomensural 
r\longa""neomensural" r\breve r1 r2 r4 r8 r16

\compressEmptyMeasures 
\override Rest.style = #'mensural
r\longa""mensural" r\breve r1 r2 r4 r8 r16 s \break 
\override Rest.style = #'neomensural 
r\longa""neomensural" r\breve r1 r2 r4 r8 r16

\compressEmptyMeasures 
\override Rest.style = #'mensural
r\longa""mensural" r\breve r1 r2 r4 r8 r16 s \break 
\override Rest.style = #'neomensural 
r\longa""neomensural" r\breve r1 r2 r4 r8 r16

\compressEmptyMeasures 
\override Rest.style = #'mensural
r\longa""mensural" r\breve r1 r2 r4 r8 r16 s \break 
\override Rest.style = #'neomensural 
r\longa""neomensural" r\breve r1 r2 r4 r8 r16

\compressEmptyMeasures 
\override Rest.style = #'mensural
r\longa""mensural" r\breve r1 r2 r4 r8 r16 s \break 
\override Rest.style = #'neomensural 
r\longa""neomensural" r\breve r1 r2 r4 r8 r16
There are no 32nd and 64th rests specifically for the mensural or neo-mensural styles. Rests from the default style are used.

**See also**


Notation Reference: [Rests], page 59.

Snippets: Section “Ancient notation” in Snippets.

**Known issues and warnings**

The glyph for the maxima rest in mensural style is actually a perfect longa rest; use two (or three) longa rests to print a maxima rest. Longa rests are not grouped automatically, so have to be done manually by using pitched rests.

**Mensural accidentals and key signatures**

The mensural style provides a sharp and a flat sign different from the default style. Mensural notation rarely used a natural sign: instead the appropriate sharp or flat is used. For example, a B natural in the key of F major would be indicated with a sharp. However, if specifically called for, the natural sign is taken from the vaticana style.

\[\text{mensural}\]

\[\text{\textbf{b} x}\]

The style for accidentals and key signatures is controlled by the glyph-name-alist property of the grobs Accidental and KeySignature, respectively; e.g.:

```latex
\override Staff.Accidental.glyph-name-alist = 
#alteration-mensural-glyph-name-alist
```

**See also**


Notation Reference: Section 1.1 [Pitches], page 1, [Accidentals], page 6, [Automatic accidentals], page 29, [Key signature], page 22.

Internals Reference: Section “KeySignature” in Internals Reference.

**Annotational accidentals (musica ficta)**

In European music from before about 1600, singers were expected to chromatically alter notes at their own initiative according to certain rules. This is called musica ficta. In modern transcriptions, these accidentals are usually printed over the note.

Support for such suggested accidentals is included, and can be switched on by setting suggestAccidentals to true.

```latex
\relative {
    fis' gis
    \set suggestAccidentals = ##t
    ais bis
}
```

\[\text{\textbf{\#\#\#}}\]
This will treat every subsequent accidental as *musica ficta* until it is unset with `\set suggestAccidentals = ##f`. A more practical way is to use `\once \set suggestAccidentals = ##t`, which can even be defined as a convenient shorthand:

\begin{verbatim}
\ficta = { \once \set suggestAccidentals = ##t }
\score { \relative 
\new MensuralVoice { 
  \once \set suggestAccidentals = ##t 
  bes'4 a2 g2 \ficta fis8 \ficta e! fis2 g1 
} }
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Example of ficta with \ficta = \{ \once \set suggestAccidentals = ##t \}}
\end{figure}

**See also**


**White mensural ligatures**

There is limited support for white mensural ligatures. To engrave white mensural ligatures, in the layout block, replace the `Ligature_bracket_engraver` with the `Mensural_ligature_engraver` in the `Voice` context:

\begin{verbatim}
\layout {
  \context { 
    \Voice 
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
  } }
\end{verbatim}

There is no additional input language to describe the shape of a white mensural ligature. The shape is rather determined solely from the pitch and duration of the enclosed notes. While this approach may take a new user a while to get accustomed to, it has the great advantage that the full musical information of the ligature is known internally. This is not only required for correct MIDI output, but also allows for automatic transcription of the ligatures.

At certain places two consecutive notes can be represented either as two squares or as an oblique parallelogram (flexa shape). In such cases the default is the two squares, but a flexa can be required by setting the `ligature-flexa` property of the second note head. The length of a flexa can be set by the note head property `flexa-width`.

For example,

\begin{verbatim}
\score { 
  \relative { 
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c''\maxima g \]
  }
\end{verbatim}
\[ d'\longa \\
/override NoteHead.ligature-flexa = ##t \\
/once \override NoteHead.flexa-width = #3.2 \\
c\breve f e d \] \\
/ [ c\maxima d\longa ] \\
/ [ e1 a, g\breve ] \\
} \\
\} \\
} \\
\} \\
} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
The following table shows all Gregorian clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. Still, you can manually force a clef glyph to be typeset on an arbitrary line, as described in [Clef], page 17. The note printed to the right side of each clef in the example column denotes the c' with respect to that clef.

<table>
<thead>
<tr>
<th>Description</th>
<th>Supported Clefs</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editio Vaticana style do clef</td>
<td>vaticana-do1, vaticana-do2, vaticana-do3</td>
<td></td>
</tr>
<tr>
<td>Editio Vaticana style fa clef</td>
<td>vaticana-fa1, vaticana-fa2</td>
<td></td>
</tr>
<tr>
<td>Editio Medicaea style do clef</td>
<td>medicaea-do1, medicaea-do2, medicaea-do3</td>
<td></td>
</tr>
<tr>
<td>Editio Medicaea style fa clef</td>
<td>medicaea-fa1, medicaea-fa2</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2: Specialist notation

hufnagel style do clef  
| hufnagel-do1, hufnagel-do2,  |
| hufnagel-do3               |

hufnagel style fa clef  
| hufnagel-fa1, hufnagel-fa2 |

hufnagel style combined do/fa clef  
| hufnagel-do-fa |

See also
Notation Reference: [Clef], page 17.

Gregorian accidentals and key signatures
Accidentals for the three different Gregorian styles are available:

vaticana medicaea hufnagel

As shown, not all accidentals are supported by each style. When trying to access an un supported accidental, LilyPond will switch to a different style.

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental.glyph-name-alist =
#alteration-mensural-glyph-name-alist
```

See also
Notation Reference: Section 1.1 [Pitches], page 1, [Accidentals], page 6, [Automatic accidentals], page 29, [Key signature], page 22.
Internals Reference: Section “KeySignature” in Internals Reference.

Divisiones
There are no rests in Gregorian chant notation; instead, it uses [Divisiones], page 459.

A divisio (plural: divisiones; Latin word for ‘division’) is a staff context symbol that is used to indicate the phrase and section structure of Gregorian music. The musical meaning of divisio minima, divisio maior, and divisio maxima can be characterized as short, medium, and long pause, somewhat like the breath marks from [Breath marks], page 138. The finalis sign not only marks the end of a chant, but is also frequently used within a single antiphonal/responsorial chant to mark the end of each section.

To use divisiones, include the file `gregorian.ly`. It contains definitions that you can apply by just inserting \divisioMinima, \divisioMaior, \divisioMaxima, and \finalis at proper places in the input. Some editions use \virgula or \caesura instead of divisio minima. Therefore, `gregorian.ly` also defines \virgula and \caesura
Predefined commands
\virgula, \caesura, \divisioMinima, \divisioMaior, \divisioMaxima, \finalis.

See also
Notation Reference: [Breath marks], page 138.
Installed Files: ly/gregorian.ly.

Gregorian articulation signs
In addition to the standard articulation signs described in section [Articulations and ornamentations], page 122, articulation signs specifically designed for use with notation in Editio Vaticana style are provided.

\include "gregorian.ly"
\score {
\new VaticanaVoice {
\override TextScript.font-family = #'typewriter
\override TextScript.font-shape = #'upright
\override Script.padding = #-0.1
\ictus_ictus " \bar "" \break
\circulus_circulus " \bar "" \break
\semicirculus_semicirculus " \bar "" \break
\accentus_accentus " \bar "" \break
\episema_episemInitium \pes b \flexa a b \episemFinis \flexa a \}
}\}

\virgula
\caesura
\ictus
\circulus
\semicirculus
\accentus
\episema
See also

Notation Reference: [Articulations and ornamentations], page 122.
Snippets: Section “Ancient notation” in Snippets.

Known issues and warnings

Some articulations are vertically placed too closely to the corresponding note heads.

Augmentum dots (morae)

Augmentum dots, also called morae, are added with the music function \augmentum. Note that \augmentum is implemented as a unary music function rather than as head prefix. It applies to the immediately following music expression only. That is, \augmentum \virga c will have no visible effect. Instead, say \virga \augmentum c or \augmentum {\virga c}. Also note that you can say \augmentum {a g} as a shortcut for \augmentum a \augmentum g.

\include "gregorian.ly"
\score {
    \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
    }
}

See also

Notation Reference: [Breath marks], page 138.
Internals Reference: Section “BreathingSign” in Internals Reference.
Snippets: Section “Ancient notation” in Snippets.

Gregorian square neume ligatures

There is limited support for Gregorian square neumes notation (following the style of the Editio Vaticana). Core ligatures can already be typeset, but essential issues for serious typesetting are still lacking, such as (among others) horizontal alignment of multiple ligatures, lyrics alignment, and proper handling of accidentals.

The support for Gregorian neumes is enabled by \includeing gregorian.ly at the beginning of the file. This makes available a number of extra commands to produce the neume symbols used in plainchant notation.

- The shape of the note head can be modified by prefixing the note name with any of the following commands: \virga, \stropha, \inclinatum, \auctum, \descendens, \ascendens, \oriscus, \quilisma, \deminutum, \cavum, \linea.
- Ligatures, properly speaking (i.e., notes joined together), are produced by placing one of the joining commands \pes or \flexa, for upwards and downwards movement, respectively, between the notes to be joined.
A note name without any qualifiers will produce a *punctum*. All other neumes, including the single-note neumes with a different shape such as the *virga*, are in principle considered as ligatures and should therefore be placed between \[ ... \].

**Single-note neumes:**

- The *punctum* is the basic note shape (in the *Vaticana* style: a square with some curvation for typographical finesse). In addition to the regular *punctum*, there is also the oblique *punctum inclinatum*, produced with the prefix *inclinatum*. The regular *punctum* can be modified with *cavum*, which produces a hollow note, and *linea*, which draws vertical lines on either side of the note.
- The *virga* has a descending stem on the right side. It is produced by the modifier *virga*.

**Ligatures**

Unlike most other neumes notation systems, the typographical appearance of ligatures is not directly dictated by the input commands, but follows certain conventions dependent on musical meaning. For example, a three-note ligature with the musical shape low-high-low, such as \[ \pes b \flexa g \], produces a Torculus consisting of three Punctum heads, while the shape high-low-high, such as \[ a \flexa g \pes b \], produces a Porrectus with a curved flexa shape and only a single Punctum head. There is no command to explicitly typeset the curved flexa shape; the decision of when to typeset a curved flexa shape is based on the musical input. The idea of this approach is to separate the musical aspects of the input from the notation style of the output. This way, the same input can be reused to typeset the same music in a different style of Gregorian chant notation.

**Liquescent neumes**

Another main category of notes in Gregorian chant is the so-called liquescent neumes. They are used under certain circumstances at the end of a syllable which ends in a 'liquescent' letter, i.e., the sounding consonants that can hold a tone (the nasals, l, r, v, j, and their diphthong equivalents). Thus, the liquescent neumes are never used alone (although some of them can be produced), and they always fall at the end of a ligature.

Liquescent neumes are represented graphically in two different, more or less interchangeable ways: with a smaller note or by ‘twisting’ the main note upwards or downwards. The first is produced by making a regular *pes* or *flexa* and modifying the shape of the second note: \[ a \pes \deminutum b \], the second by modifying the shape of a single-note neume with *auctum* and one of the direction markers *descendens* or *ascendens*, e.g., \[ \auctum \descendens a \].

**Special signs**

A third category of signs is made up of a small number of signs with a special meaning (which, incidentally, in most cases is only vaguely known): the *quilisma*, the *oriscus*, and the *strophicus*. These are all produced by prefixing a note name with the corresponding modifier, \*quilisma*, \*oriscus*, or \*strophic*.

Virtually, within the ligature delimiters \[ and \], any number of heads may be accumulated to form a single ligature, and head prefixes like \*pes*, \*flexa*, \*virga*, \*inclinatum*, etc., may be mixed in as desired. The use of the set of rules that underlies the construction of the ligatures in the above table is accordingly extrapolated. This way, infinitely many different ligatures can be created.

Note that the use of these signs in the music itself follows certain rules, which are not checked by LilyPond. E.g., the *quilisma* is always the middle note of an ascending ligature, and usually falls on a half-tone step, but it is perfectly possible, although incorrect, to make a single-note quilisma.

In addition to the note signs, *gregorian.ly* also defines the commands \*versus*, \*responsum*, \*ij*, \*iij*, \*IJ*, and \*IIJ*, that will produce the corresponding characters, e.g.,
for use in lyrics, as section markers, etc. These commands use special Unicode characters and will only work if a font is used which supports them.

The following table shows a limited, but still representative pool of Gregorian ligatures, together with the code fragments that produce the ligatures. The table is based on the extended neumes table of the 2nd volume of the Antiphonale Romanum (*Liber Hymnarius*), published 1983 by the monks of Solesmes. The first column gives the name of the ligature, with the main form in boldface and the liquescent forms in italics. The third column shows the code fragment that produces this ligature, using `g`, `a`, and `b` as example pitches.

**Single-note neums**

<table>
<thead>
<tr>
<th>Basic and Liquescent forms</th>
<th>Output</th>
<th>LilyPond code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Punctum</strong></td>
<td></td>
<td>[ b ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ \cavum b ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ \linea b ]</td>
</tr>
<tr>
<td><strong>Punctum Auctum Ascendens</strong></td>
<td></td>
<td>[ \auctum \ascendens b ]</td>
</tr>
<tr>
<td><strong>Punctum Auctum Descendens</strong></td>
<td></td>
<td>[ \auctum \descendens b ]</td>
</tr>
<tr>
<td><strong>Punctum inclinatum</strong></td>
<td></td>
<td>[ \inclinatum b ]</td>
</tr>
<tr>
<td><strong>Punctum Inclinatum Auctum</strong></td>
<td></td>
<td>[ \inclinatum \auctum b ]</td>
</tr>
</tbody>
</table>
\textit{Punctum Inclinatum Parvum}

\[
\inclinatum \deminutum b
\]

\textit{Virga}

\[
\text{\textbullet}
\]

\textit{Two-note ligatures}

\textit{Clivis vel Flexa}

\[
\flexa g
\]

\textit{Clivis Aucta Descendens}

\[
\flexa \auctum \descendens g
\]

\textit{Clivis Aucta Ascendens}

\[
\flexa \auctum \ascendens g
\]

\textit{Cephalicus}

\[
\flexa \deminutum g
\]

\textit{Podatus/Pes}

\[
\pes b
\]

\textit{Pes Auctus Descendens}

\[
\pes \auctum \descendens b
\]

\textit{Pes Auctus Ascendens}

\[
\pes \auctum \ascendens b
\]
Epiphonus
\[ g \pes \deminutum b \]

Pes Initio Debilis
\[ \deminutum g \pes b \]

Pes Auctus Descendens Initio Debilis
\[ \deminutum g \pes \auctum \descendens b \]

Multi-note ligatures

Torculus
\[ a \pes b \flexa g \]

Torculus Auctus Descendens
\[ a \pes b \flexa \auctum \descendens g \]

Torculus Deminutus
\[ a \pes b \flexa \deminutum g \]

Torculus Initio Debilis
\[ \deminutum a \pes b \flexa g \]

Torculus Auctus Descendens Initio Debilis
\[ \deminutum a \pes b \flexa \auctum \descendens g \]

Torculus Deminutus Initio Debilis
\[ \deminutum a \pes b \flexa \deminutum g \]
**Porrectus**
\[
\begin{align*}
\text{\[ a \flexa g \pes b \]} \\
\end{align*}
\]

**Porrectus Auctus Descendens**
\[
\begin{align*}
\text{\[ a \flexa g \pes \auctum \descendens b \]} \\
\end{align*}
\]

**Porrectus Deminutus**
\[
\begin{align*}
\text{\[ a \flexa g \pes \deminutum b \]} \\
\end{align*}
\]

**Climacus**
\[
\begin{align*}
\text{\[ \virga b \inclinatum a \inclinatum g \]} \\
\end{align*}
\]

**Climacus Auctus**
\[
\begin{align*}
\text{\[ \virga b \inclinatum a \inclinatum \auctum g \]} \\
\end{align*}
\]

**Climacus Deminutus**
\[
\begin{align*}
\text{\[ \virga b \inclinatum a \inclinatum \deminutum g \]} \\
\end{align*}
\]

**Scandicus**
\[
\begin{align*}
\text{\[ g \pes a \virga b \]} \\
\end{align*}
\]

**Scandicus Auctus Descendens**
\[
\begin{align*}
\text{\[ g \pes a \pes \auctum \descendens b \]} \\
\end{align*}
\]

**Scandicus Deminutus**
\[
\begin{align*}
\text{\[ g \pes a \pes \deminutum b \]} \\
\end{align*}
\]

**Special Signs**
**Quilisma**

\[
\text{\quilisma a \pes b}
\]

**Quilisma Pes Auctus Descendens**

\[
\text{\quilisma \pes \auctum \descendens b}
\]

**Oriscus**

\[
\text{\oriscus b}
\]

**Pes Quassus**

\[
\text{\oriscus \pes \virga b}
\]

**Pes Quassus Auctus Descendens**

\[
\text{\oriscus \pes \auctum \descendens b}
\]

**Salicus**

\[
\text{g \oriscus a \pes \virga b}
\]

**Salicus Auctus Descendens**

\[
\text{g \oriscus a \pes \auctum \descendens b}
\]

**(Apo)stropha**

\[
\text{\stropha b}
\]

**Stropha Aucta**

\[
\text{\stropha \auctum b}
\]

**Bistropha**

\[
\text{\stropha b \stropha b}
\]
Tristropha

\[
\text{stroph a b stroph a b stroph a b}
\]

Trigonus

\[
\text{stroph a b stroph a b stroph a a}
\]

Predefined commands

The following head prefixes are supported: \virga, \stropha, \inclinatum, \auctum, \descendens, \ascendens, \oriscus, \quilisma, \deminutum, \cavum, \linea. Head prefixes can be accumulated, though restrictions apply. For example, either \descendens or \ascendens can be applied to a head, but not both to the same head.

Two adjacent heads can be tied together with the \pes and \flexa infix commands for a rising and falling line of melody, respectively.

Use the unary music function \augmentum to add augmentum dots.

See also

Notation Reference: [Gregorian square neume ligatures], page 461, [White mensural ligatures], page 456, [Ligatures], page 448.

Known issues and warnings

When an \augmentum dot appears at the end of the last staff within a ligature, it is sometimes vertically placed wrong. As a workaround, add an additional skip note (e.g., s8) as last note of the staff.

\augmentum should be implemented as a head prefix rather than a unary music function, such that \augmentum can be intermixed with head prefixes in arbitrary order.

2.9.5 Typesetting Kievan square notation

Kievan contexts

As with Mensural and Gregorian notation, the predefined KievanVoice and KievanStaff contexts can be used to engrave a piece in square notation. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant:

\%
Font settings for Cyrillic
\paper {
  #(define fonts
    (set-global-fonts
      #:roman "Linux Libertine O,serif"
    ))
}
\score {
  <<
    \new KievanVoice = "melody" \relative c' {

\cadenza
\c4 c c c c2 b \longa
\bar "k"
}
\new Lyrics \lyricsto "melody" {
    -- -- -- -- .
}
}>>
}

See also

Known issues and warnings
LilyPond supports Kievan notation of the Synodal style, as used in the corpus of chantbooks printed by the Russian Holy Synod in the 1910’s and recently reprinted by the Moscow Patriarchate Publishing House. LilyPond does not support the older (less common) forms of Kievan notation that were used in Galicia to notate Rusyn plainchant.

Kievan clefs
There is only one clef used in Kievan notation (the Tse-fa-ut Clef). It is used to indicate the position of c:
\clef "kievan-do"
\kievanOn
\c'\

See also
Notation Reference: [Clef], page 17.

Kievan notes
For Kievan square notation, the appropriate note head style needs to be chosen and the flags and stems need to be turned off. This is accomplished by calling the \kievanOn function, which sets the appropriate properties of the note head, stems, and flags. Once Kievan note heads are not needed, these properties can be reverted by calling the \kievanOff function.

The Kievan final note, which usually comes at the end of a piece of music, may be selected by setting the duration to \longa. The Kievan recitative mark, used to indicate the chanting of several syllables on one note, may be selected by setting the duration to \breve. The following example demonstrates the various Kievan note heads:
\autoBeamOff
\cadenza

Chapter 2: Specialist notation

\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2

See also


Notation Reference: Section A.9 [Note head styles], page 705.

Known issues and warnings

LilyPond automatically determines if the stem up or stem down form of a note is drawn. When setting chant in square notation, however, it is customary to have the stems point in the same direction within a single melisma. This can be done manually by setting the *direction* property of the *Stem* object.

Kievan accidentals

The *kievan* style for accidentals is selected with the *glyph-name-alist* property of the grob *Accidental*. The *kievan* style provides a sharp and a flat sign different from the default style. There is no natural sign in Kievan notation. The sharp sign is not used in Synodal music but may occur in earlier manuscripts. It has been included primarily for the sake of compatibility.

\clef "kievan-do"
\override Accidental.glyph-name-alist =
#alteration-kievan-glyph-name-alist
bes' dis'

See also


Notation Reference: [Accidentals], page 6, [Automatic accidentals], page 29, Section A.8 [The Emmentaler font], page 683,

Kievan bar line

A decorative figure is commonly placed at the end of a piece of Kievan notation, which may be called the Kievan final bar line. It can be invoked as \bar "k".

\kievanOn
\clef "kievan-do"
c' \bar "k"
See also

Notation Reference: Section 1.2.5 [Bars], page 99, Section A.8 [The Emmentaler font], page 683.

Kievan melismata

Notes within a Kievan melisma are usually placed close to each other and the melismata separated by whitespace. This is done to allow the chanter to quickly identify the melodic structures of Znamenny chant. In LilyPond, melismata are treated as ligatures and the spacing is implemented by the `Kievan_ligature_engraver`.

When the `KievanVoice` and `KievanStaff` contexts are used, the `Kievan_ligature_engraver` is enabled by default. In other contexts, it can be invoked by replacing the `Ligature_bracket_engraver` with the `Kievan_ligature_engraver` in the layout block:

```lilypond
\layout {
  \context { 
    \Voice 
    \remove "Ligature_bracket_engraver"
    \consists "Kievan_ligature_engraver"
  }
}
```

The spacing between the notes within a Kievan ligature can be controlled by setting the `padding` property of the `KievanLigature`.

The following example demonstrates the use of Kievan ligatures:

```lilypond
% Font settings for Cyrillic
\paper { #(define fonts (set-global-fonts #:roman "Linux Libertine O,serif" )) }
\score { << \new KievanVoice = "melody" \relative c' { \cadenzaOn e2 \[ e4( d4 ) \] \[ c4( d e d ) \] e1 \bar "k" } \new Lyrics \lyricsto "melody" { -- -- -- } >> } 
```

See also

known issues and warnings

Horizontal spacing of ligatures is poor.

2.9.6 Working with ancient music—scenarios and solutions

Working with ancient music frequently involves particular tasks which differ considerably from the modern notation for which LilyPond is designed. In the rest of this section, a number of typical scenarios are outlined, with suggestions of solutions. These involve:

- how to make incipits (i.e., prefatory material to indicate what the original has looked like) to modern transcriptions of mensural music;
- how to achieve the Mensurstriche layout frequently used for modern transcriptions of polyphonic music;
- how to transcribe Gregorian chant in modern notation;
- how to generate both ancient and modern notation from the same source.

Incipits

It is customary when transcribing mensural music into modern notation to place an indication of how the initial rests and note or notes of the original version appeared—including the original clefs. This is called an incipit. The \incipit command uses the indent of the main staff to set the width occupied by the incipit, and incipit-width to set the width of the incipit staff.

\score {
\set Staff.instrumentName = "Tenor"
\override Staff.InstrumentName.self-alignment-X = #RIGHT
\incipit { \clef "mensural-c4" \key f \major r\breve r1 c'1 }
\clef "treble_8"
\key f \major
R1 r2 c'2 |
a4. c'8
}
\new Lyrics \lyricsto Tenor { Cyn -- thia your }

\layout
{
indent = 5\cm
incipit-width = 3\cm
}
}

\Staffs{\Tenor { Cyn-thia your

Known issues and warnings

Note that instrumentName must be set in the music for the incipit to be produced. If no instrument name is required then use \set Staff.instrumentName = ".".
Mensurstriche layout

*Mensurstriche* (‘mensuration lines’) is the accepted term for bar lines that are drawn between the staves of a system but not through the staves themselves. It is a common way to preserve the rhythmic appearance of the original, i.e., not having to break syncopated notes at bar lines, while still providing the orientation aids that bar lines give.

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a *StaffGroup* instead of a *ChoirStaff*. The bar line on staves is blanked out using \hide.

```latex
\global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}
```

```latex
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```

Transcribing Gregorian chant

Gregorian chant can be transcribed into modern notation with a number of simple tweaks.

**Stems.** Stems can be left out altogether by \remove-ing the *Stem_engraver* from the Voice context:

```latex
\layout {
  ...
  \context {
    \Voice
    \remove "Stem_engraver"
  }
}
```

**Timing.** For unmetered chant, there are several alternatives.

The *Time_signature_engraver* can be removed from the Staff context without any negative side effects. The alternative, to make it transparent, will leave an empty space in the score, since the invisible signature will still take up space.

In many cases, \set Score.timing = ##f will give good results. Another alternative is to use \cadenzaOn and \cadenzaOff.

To remove the bar lines, the radical approach is to \remove the *Bar_engraver* from the Staff context. Again, one may want to use \hide *BarLine* instead, if an occasional bar line is wanted.
A common type of transcription is recitativic chant where the repeated notes are indicated with a single breve. The text to the recitation tone can be dealt with in two different ways: either set as a single, left-aligned syllable:

\include "gregorian.ly"

chant = \relative {
  \clef "G_8"
  c\breve c4 b4 a c2 c4 \divisioMaior
  c\breve c4 c f, f \finalis
}

verba = \lyricmode {
  \once \override LyricText.self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText.self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {\Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Voice
      \remove "Stem_engraver"
    }
  }
}

\include "gregorian.ly"

chant = \relative {

\hspace{2in}

Noctem quietam et finem perfectum concedat nobis Dominus omnipo-

tens.

This works fine, as long as the text doesn’t span a line break. If that is the case, an alternative is to add hidden notes to the score, as below.

In some transcription styles, stems are used occasionally, for example to indicate the transition from a single-tone recitative to a fixed melodic gesture. In these cases, one can use either \hide Stem or \override Stem.length = #0 instead of \remove-ing the Stem_engraver and restore the stem when needed with the corresponding \undo \hide Stem.
Another common situation is transcription of neumatic or melismatic chants, i.e., chants with a varying number of notes to each syllable. In this case, one would want to set the syllable groups clearly apart, usually also the subdivisions of a longer melisma. One way to achieve this is to use a fixed \time, e.g., 1/4, and let each syllable or note group fill one of these measures, with the help of tuplets or shorter durations. If the bar lines and all other rhythmical indications are made transparent, and the space around the bar lines is increased, this will give a fairly good representation in modern notation of the original.

To avoid that syllables of different width (such as “-ri” and “-rum”) spread the syllable note groups unevenly apart, the 'X-extent property of the LyricText object may be set to a fixed value. Another, more cumbersome way would be to add the syllables as \markup elements. If further adjustments are necessary, this can be easily done with a ‘notes’.
spirLyr = \lyricmode {
  Spi -- ri -- _ tus _ Do -- mi -- ni _ re -- ple -- _ vit _
  or -- _ bem _ ter -- ra -- _ rum, al -- _ le -- _ lu
  -- _ ia.
}
\score {
\new Staff <<
  \new Voice = "chant" \spiritus
  \new Lyrics = "one" \lyricsto "chant" \spirLyr
>>
\layout {
  \context {
    \Staff
    \remove "Time_signature_engraver"
    \override BarLine.X-extent = #'(-1 . 1)
    \hide Stem
    \hide Beam
    \hide BarLine
    \hide TupletNumber
  }
}

Ancient and modern from one source

Using tags to produce mensural and modern music from the same source

By using tags, it's possible to use the same music to produce both mensural and modern music. In this snippet, a function \texttt{menrest} is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position. Tags are used to produce different types of bar line at the end of the music, but tags can also be used where other differences are needed: for example using “whole measure rests” (R1, R\breve etc.) in modern music, but normal rests (r1, r\breve, etc.) in the mensural version. Note that converting mensural music to its modern equivalent is usually referred to as \texttt{transcription}.

\begin{verbatim}
menrest = #(define-music-function (note)
(l:ly:music?)
#
  \tag #'mens $(make-music 'RestEvent note)
  \tag #'mod $(make-music 'RestEvent note 'pitch '())
#)}
\end{verbatim}
MensStyle = {
\autoBeamOff
\override NoteHead.style = #'petrucci
\override Score.BarNumber.transparent = ##t
\override Stem.neutral-direction = #up
}

finalis = {
\once \override BreathingSign.stencil = #ly:breathing-sign::finalis
\once \override BreathingSign.Y-offset = #0
\once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
\once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

\breathe
}

Music = \relative c' {  
\set Score.tempoHideNote = ##t  
\key f \major  
\time 4/4  
g\1 \d'2 \menrest bes4 bes2 a2 r4 g4 fis2.  
\tag #'mens { \finalis }  
\tag #'mod { \bar "||" }  
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {  
\keepWithTag #'mens {  
<<  
\new MensuralStaff  
{  
\new MensuralVoice = Cantus \clef "mensural-c1" \MensStyle \Music  
}  
\new Lyrics \lyricsto Cantus \MenLyr  
>>  
}  
}

\score {  
\keepWithTag #'mod {  
\new ChoirStaff <<  
\new Staff  
{  
\new Voice = Sop \with {  
\remove "Note_heads_engraver"  
\consists "Completion_heads_engraver"  
\remove "Rest_engraver"  
\consists "Completion_rest_engraver"  
}  
{  
\shiftDurations #1 #0 { \autoBeamOff \Music  
}  
}  
}  
}
Editorial markings

2.10 World music

The purpose of this section is to highlight musical notation issues that are relevant to traditions outside the Western tradition.

2.10.1 Common notation for non-Western music

This section discusses how to enter and print music scores that do not belong to the Western classical tradition, also referred to as Common Practice Period.

Extending notation and tuning systems

Standard classical notation (also known as Common Practice Period notation) is commonly used in all sorts of music, not limited to ‘classical’ Western music. This notation is discussed in Section 1.1.1 [Writing pitches], page 1, and the various note names that may be used are explained in [Note names in other languages], page 8.

Some types of non-Western music and folk/traditional music often employ alternative or extended tuning systems that do not fit easily into standard, classical notation.

Standard notation is still used but with pitch differences being implicit. For example, Arabic music is notated with semi and quarter-tone accidentals but with precise pitch alterations being determined by context. In the case of Arabic music, the init file arabic.ly provides a suitable set of macros and definitions that extend the standard notation using Italian note names. For more details see Section 2.10.2 [Arabic music], page 479.

Other types of music require extended or unique notations, for example, Turkish classical music (also known as Ottoman classical music) employs melodic forms known as makamlar where intervals are based on 1/9 divisions of the whole tone. Standard, Western staff notes are still used, but with special accidentals uniquely defined in the files turkish-makam.ly. For more information on Turkish classical music and makamlar see Section 2.10.3 [Turkish classical music], page 484.

Other, related init files are also available; hel-arabic.ly and makam.ly.

To locate these init files on your system, see Section “Other sources of information” in Learning Manual.
See also


Notation Reference: Section 1.1.1 [Writing pitches], page 1, [Note names in other languages], page 8, Section 2.10.2 [Arabic music], page 479, Section 2.10.3 [Turkish classical music], page 484.

2.10.2 Arabic music

This section highlights issues that are relevant to notating Arabic music.

References for Arabic music

Arabic music so far has been mainly an oral tradition. When music is transcribed, it is usually in a sketch format, on which performers are expected to improvise significantly. Increasingly, Western notation, with a few variations, is adopted in order to communicate and preserve Arabic music.

Some elements of Western musical notation such as the transcription of chords or independent parts, are not required to typeset the more traditional Arabic pieces. There are however some different issues, such as the need to indicate medium intervals that are somewhere between a semi-tone and a tone, in addition to the minor and major intervals that are used in Western music. There is also the need to group and indicate a large number of different maqams (modes) that are part of Arabic music.

In general, Arabic music notation does not attempt to precisely indicate microtonal elements that are present in musical practice.

Several issues that are relevant to Arabic music are covered elsewhere:

• Note names and accidentals (including quarter tones) can be tailored as discussed in Section 2.10.1 [Common notation for non-Western music], page 478.
• Additional key signatures can also be tailored as described in [Key signature], page 22.
• Complex time signatures may require that notes be grouped manually as described in [Manual beams], page 96.
• Takasim which are rhythmically free improvisations may be written down omitting bar lines as described in [Unmetered music], page 75.

See also

Notation Reference: Section 2.10.1 [Common notation for non-Western music], page 478, [Key signature], page 22, [Manual beams], page 96.

Snippets: Section “World music” in Snippets.

Arabic note names

Traditional Arabic note names can be quite long and so may not always be suitable for the purpose of music writing.

The hel-arabic.ly file allows English note names to be used. This is a rast scale using hel-arabic.ly:

\include "hel-arabic.ly"
\relative {
\key c \rast
   c' d edb f | g a bdb c | c bb a g | f d c
}
The *arabic.ly* file allows Italian (or Solfege) note names to be used instead. This is a *rast* scale using *arabic.ly*;

```lilypond
\include "arabic.ly"
\relative { do' re misb fa | sol la sisb do | sisb la sol fa | misb re do }
```

“Rast” is a heptatonic scale that uses quarter-tone intervals and is considered the most important and central scale of the “Arabic Maqamat”. For the full list of supported Arabic scales please refer to either the *hel-arabic.ly* or *arabic.ly* files that are both included with LilyPond.

The use of standard Western notation to notate non-Western music is discussed in Section 2.10.1 [Common notation for non-Western music], page 478. Also see [Note names in other languages], page 8.

The symbol for semi-flat does not match the symbol which is used in Arabic notation. The \dwn symbol defined in *arabic.ly* may be used preceding a flat symbol as a workaround if it is important to use the specific Arabic semi-flat symbol. The appearance of the semi-flat symbol in the key signature cannot be altered by using this method.

```lilypond
\include "arabic.ly"
\relative { \set Staff.extraNatural = ##f dod' dob dosd \dwn dob dobsb dodsd do do }
```

See also

Notation Reference: [Note names in other languages], page 8, Section 2.10.1 [Common notation for non-Western music], page 478, Section 3.3.1 [Including LilyPond files], page 517.

Installed Files: 1y/arabic.ly 1y/hel-arabic.ly

Snippets: Section “World music” in Snippets.

**Arabic key signatures**

In addition to the minor and major key signatures, Arabic key signatures are defined in either *hel-arabic.ly* or *arabic.ly* files and define many different maqam groups.

In general, a maqam uses the key signature of its group, or a neighbouring group, and varying accidentals are marked throughout the music. Arabic maqams only allow for limited modulations, due to the nature of Arabic musical instruments.

Here is an example of the key signature for a “maqam muhayer” piece of music:

```lilypond
\key re \bayati
```

Here *re* is the default pitch of the muhayer maqam, and *bayati* is the name of the base maqam in the group.
While the key signature indicates the group, it is common for the title to indicate the more specific maqam, so in this example, the name of "maqam muhayer" should also appear in the title.

Other maqams in the same bayati group, as shown in the table below (e.g., bayati, hussaini, saba, and ushaq) can be indicated in the same way. These are all variations of the base and most common maqam in the group, which is bayati. They usually differ from the base maqam in their upper tetrachords, or certain flow details that do not change their fundamental nature, as siblings.

The other maqam in the same group (nawa) is related to bayati by modulation and is shown in the table in parentheses for those that are modulations of their base maqam. Nawa, for example, can be indicated as follows:

\key sol \bayati

In Arabic music, the same term, for example bayati, that is used to indicate a maqam group, will also be a maqam that is usually the most important in the group so can also be thought of as a base maqam.

Here is one suggested grouping that maps the more common maqams to key signatures:

<table>
<thead>
<tr>
<th>maqam group</th>
<th>key</th>
<th>finalis</th>
<th>Other maqams in group (finalis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajam</td>
<td>major</td>
<td>sib</td>
<td>jaharka (fa)</td>
</tr>
<tr>
<td>bayati</td>
<td>bayati</td>
<td>re</td>
<td>hussaini, muhayer, saba, ushaq, nawa (sol)</td>
</tr>
<tr>
<td>hijaz</td>
<td>kurd</td>
<td>re</td>
<td>shahnaz, shad arban (sol), hijazkar (do)</td>
</tr>
<tr>
<td>iraq</td>
<td>iraq</td>
<td>sib</td>
<td>-</td>
</tr>
<tr>
<td>kurd</td>
<td>kurd</td>
<td>re</td>
<td>hijazkar kurd (do)</td>
</tr>
<tr>
<td>nahawand</td>
<td>minor</td>
<td>do</td>
<td>busalik (re), farah faza (sol)</td>
</tr>
<tr>
<td>nakriz</td>
<td>minor</td>
<td>do</td>
<td>nawa athar, hisar (re)</td>
</tr>
<tr>
<td>rast</td>
<td>rast</td>
<td>do</td>
<td>mahur, yakah (sol)</td>
</tr>
<tr>
<td>sikah</td>
<td>sikah</td>
<td>misb</td>
<td>huzam</td>
</tr>
</tbody>
</table>

**Selected Snippets**

*Non-traditional key signatures*

The commonly used \key command sets the keyAlterations property in the Staff context. To create non-standard key signatures, set this property directly.

The format of this command is a list:

\set Staff.keyAlterations = #`(((octave . step) . alter) ((octave . step) . alter) ...) \%

where, for each element in the list octave specifies the octave (0 being the octave from middle c to the b above), step specifies the note within the octave (0 means c and 6 means b), and alter is ,SHARP ,FLAT ,DOUBLE-SHARP etc.

Alternatively, using the more concise format for each item in the list, (step . alter) specifies the same alteration holds in all octaves. For microtonal scales where a “sharp” is not 100 cents, alter refers to the proportion of a 200-cent whole tone.

\include "arabic.ly"
\relative do' {
 \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
 (1 . ,SEMI-FLAT)
 (2 . ,FLAT)
 (5 . ,FLAT)
 (6 . ,SEMI-FLAT))
 %\set Staff.extraNatural = ##f
Chapter 2: Specialist notation

Arabic time signatures

Some Arabic and Turkish music classical forms such as Semai use unusual time signatures such as 10/8. This may lead to an automatic grouping of notes that is quite different from existing typeset music, where notes may not be grouped on the beat, but in a manner that is difficult to match by adjusting automatic beaming. The alternative is to switch off automatic beaming and beam the notes manually. Even if a match to existing typeset music is not required, it may still be desirable to adjust the automatic beaming behaviour and/or use compound time signatures.

Selected Snippets

Arabic improvisation

For improvisations or taqasim which are temporarily free, the time signature can be omitted and \cadenzaOn can be used. Adjusting the accidental style might be required, since the absence of bar lines will cause the accidental to be marked only once. Here is an example of what could be the start of a hijaz improvisation:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```

See also

Notation Reference: [Key signature], page 22.
Installed Files: ly/arabic.ly ly/hel-arabic.ly
Snippets: Section “World music” in Snippets, Section “Pitches” in Snippets.
Internals Reference: Section “KeySignature” in Internals Reference.
See also


Notation Reference: [Manual beams], page 96, [Automatic beams], page 84, [Unmetered music], page 75, [Automatic accidentals], page 29, [Setting automatic beam behavior], page 87, [Time signature], page 66.

Installed Files: ly/arabic.ly ly/hel-arabic.ly

Snippets: Section “World music” in Snippets.

Arabic music example

Here is a template that also uses the start of a Turkish Semai that is familiar in Arabic music education in order to illustrate some of the peculiarities of Arabic music notation, such as medium intervals and unusual modes that are discussed in this section.

```latex
\include "arabic.ly"
\score {
  \relative {  
    \set Staff.extraNatural = ##f  
    \set Staff.autoBeaming = ##f  
    \key re \bayati  
    \time 10/8  
    re'4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8  
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8  
  }
  \header {  
    title = "Semai Muhayer"  
    composer = "Jamil Bek"  
  }
}
```

See also

Installed Files: ly/arabic.ly ly/hel-arabic.ly

Snippets: Section “World music” in Snippets.

Further reading for Arabic music

There are some variations in the details of how maqams are grouped, despite agreement of grouping maqams related through common lower tetra chords or by modulation. There are also some inconsistencies, even within the same texts, on how key signatures for a particular maqam
should be specified. However, it is common to use a key signature per ‘group’ of maqams instead of individual key signatures for each maqam separately.

- There are also some websites that explain maqams and even provide audio examples:
  - https://www.maqamworld.com/
  - https://www.turath.org/
- Method books by the following authors for the Oud (the Arabic lute) contain examples of mainly Turkish and Arabic compositions.
  - Charbel Rouhana
  - George Farah
  - Ibrahim Ali Darwish Al-masri

### 2.10.3 Turkish classical music

This section highlights issues that are relevant to notating Turkish classical music.

#### References for Turkish classical music

Turkish classical music developed in the Ottoman Empire at roughly the same time as classical music in Europe, and has continued on into the 20th and 21st centuries as a vibrant and distinct tradition with its own compositional forms, theory and performance styles. Among its striking features is the use of microtonal intervals based on ‘commas’ of 1/9 of a tone, from which are constructed the melodic forms known as *makam* (plural *makamlar*) are constructed.

Some issues relevant to Turkish classical music are covered elsewhere. Special note names and accidentals are explained in Section 2.10.1 [Common notation for non-Western music], page 478.

#### Turkish note names

Pitches in Turkish classical music traditionally have unique names and the basis of pitch on 1/9-tone divisions means that makamlar employ a completely different set of intervals compared to Western scales and modes:

From a modern, notational point of view it is convenient to use standard, Western staff notes (c, d, e, etc.) but with custom accidentals that raise or lower notes by intervals of 1/9, 4/9, 5/9 or 8/9 of a tone.

These custom accidentals are defined in the file *turkish-makam.ly*.

For a more general explanation of non-Western music notation, see Section 2.10.1 [Common notation for non-Western music], page 478.

#### See also

- Notation Reference: Section 2.10.1 [Common notation for non-Western music], page 478.

#### Turkish key signatures

Lilypond supports over 200 makam key signature definitions – well beyond what is used in Turkish classical music – with each makam having its own specific tonic / finalis pitch (known as ‘karar’ in Turkish).

It is important to be aware of the finalis of each makam. Here is an example where *g* is the default tonic and *rast* is the name of the makam.

\key g \rast

The correct accidentals, koma flat (b1) and koma sharp (f4), (both in relation to the tonic g), will be displayed automatically.
Selected Snippets

Turkish Makam example

This template uses the start of a well-known Turkish Saz Semai that is familiar in the repertoire in order to illustrate some of the elements of Turkish music notation.

\%
Initialize makam settings
\include "turkish-makam.ly"

\header {
  title = "Hüseyni Saz Semaisi"
  composer = "Lavtacı Andon"
}

\relative {
  \set Staff.extraNatural = ##f
  \set Staff.autoBeaming = ##f

  \key a \huseyni
  \time 10/8

  a'4 g'16 [fb] e8. [d16] d [c d e] c [d c8] bfc |
  a16 [bfc a8] bfc c16 [d c8] d16 [e d8] e4 fb8 |
  d4 a'8 a16 [g fb e] fb8 [g] a8. [b16] a16 [g] |
  g4 g16 [fb] fb8. [e16] e [g fb e] e4 r8 |
}

Hüseyni Saz Semaisi

Lavtacı Andon

Further reading for Turkish music

- *Turk Musikisi Nazariyati ve Usulleri: Kudum Velveleleri* by Ismail Hakki Ozkan [(Kultur serisi, 41) (Turkish) Paperback – 1986]
  contains information about the theory of makams and usul.
- *Music of the Ottoman Court* by Walter Feldman [VWB Hardback – 1996]
  contains information about the history of Ottoman court music.
  contains information in English regarding Turkish makam including two CDs.
3 General input and output

This section deals with general LilyPond input and output issues, rather than specific notation.

3.1 Input structure

The main format of input for LilyPond are text files. By convention, these files end with .ly.

3.1.1 Structure of a score

A \score block must contain a single music expression delimited by curly brackets:

\score { ...
  ...
}

Note: There must be only one outer music expression in a \score block, and it must be surrounded by curly brackets.

This single music expression may be of any size, and may contain other music expressions to any complexity. All of these examples are music expressions:

\{ c'4 c' c' c' \}
\{
  \{ c'4 c' c' c' \}
  \{ d'4 d' d' d' \}
\}

\<<<
  \new Staff { c'4 c' c' c' }
  \new Staff { d'4 d' d' d' }
\>>>
Comments are one exception to this general rule. (For others see Section 3.1.5 [File structure], page 490.) Both single-line comments and comments delimited by `%{ ... %}` may be placed anywhere within an input file. They may be placed inside or outside a \texttt{\score} block, and inside or outside the single music expression within a \texttt{\score} block.

Remember that even in a file containing only a \texttt{\score} block, it is implicitly enclosed in a \texttt{\book} block. A \texttt{\book} block in a source file produces at least one output file, and by default the name of the output file produced is derived from the name of the input file, so \texttt{fandangofoelephants.ly} will produce \texttt{fandangofoelephants.pdf}.

(For more details about \texttt{\book} blocks, see Section 3.1.2 [Multiple scores in a book], page 487, Section 3.1.3 [Multiple output files from one input file], page 488, Section 3.1.5 [File structure], page 490.)

See also


3.1.2 Multiple scores in a book

A document may contain multiple pieces of music and text. Examples of these are an etude book, or an orchestral part with multiple movements. Each movement is entered with a \texttt{\score} block,

\begin{verbatim}
\score {
    ...music...
}
\end{verbatim}

and texts are entered with a \texttt{\markup} block,

\begin{verbatim}
\markup {
    ...text...
}
\end{verbatim}

All the movements and texts which appear in the same .ly file will normally be typeset in the form of a single output file.

\begin{verbatim}
\score {
    ...
}
\markup {
    ...
}
\score {
    ...
}
\end{verbatim}

One important exception is within lilypond-book documents, where you explicitly have to add a \texttt{\book} block, otherwise only the first \texttt{\score} or \texttt{\markup} will appear in the output.

The header for each piece of music can be put inside the \texttt{\score} block. The \texttt{piece} name from the header will be printed before each movement. The title for the entire book can be put inside the \texttt{\book}, but if it is not present, the \texttt{\header} which is at the top of the file is inserted.

\begin{verbatim}
\header {
    title = "Eight miniatures"
    composer = "Igor Stravinsky"
}
\end{verbatim}
Pieces of music may be grouped into book parts using \bookpart blocks. Book parts are separated by a page break, and can start with a title, like the book itself, by specifying a \header block.

\bookpart {
  \header {
    title = "Book title"
    subtitle = "First part"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Second part"
  }
  \score { ... }
  ...
}

### 3.1.3 Multiple output files from one input file

If you want multiple output files from the same .ly file, then you can add multiple \book blocks, where each such \book block will result in a separate output file. If you do not specify any \book block in the input file, LilyPond will implicitly treat the whole file as a single \book block, see Section 3.1.5 [File structure], page 490.

When producing multiple files from a single source file, LilyPond ensures that none of the output files from any \book block overwrites the output file produced by a preceding \book from the same input file.

It does this by adding a suffix to the output name for each \book which uses the default output file name derived from the input source file.

The default behaviour is to append a version-number suffix for each name which may clash, so

\book {
  \score { ... }
  \paper { ... }
}
\book {
3.1.4 Output file names

LilyPond provides facilities to allow you to control what file names are used by the various back-ends when producing output files.

In the previous section, we saw how LilyPond prevents name-clashes when producing several outputs from a single source file. You also have the ability to specify your own suffixes for each \book block, so for example you can produce files called eightminiatures-Romanze.pdf, eightminiatures-Menuetto.pdf and eightminiatures-Nocturne.pdf by adding a \bookOutputSuffix declaration inside each \book block.

\book {
  \bookOutputSuffix "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \paper { ... }
}

You can also specify a different output filename for book block, by using \bookOutputName declarations

\book {
  \bookOutputName "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
  \paper { ... }
}
The file above will produce these output files:
- Romanze.pdf,
- Menuetto.pdf and
- Nocturne.pdf.

### 3.1.5 File structure

A .ly file may contain any number of toplevel expressions, where a toplevel expression is one of the following:

- An output definition, such as \paper, \midi, and \layout. Such a definition at the toplevel changes the default book-wide settings. If more than one such definition of the same type is entered at the top level the definitions are combined, but in conflicting situations the later definitions take precedence. For details of how this affects the \layout block see Section 4.2.1 [The \layout block], page 557.

- A direct scheme expression, such as #(set-default-paper-size "a7" 'landscape) or #(ly:set-option 'point-and-click #f).

- A \header block. This sets the global (i.e., the top of file) header block. This is the block containing the default settings of titling fields like composer, title, etc., for all books within the file (see [Titles explained], page 492).

- A \score block. This score will be collected with other toplevel scores, and combined as a single \book. This behavior can be changed by setting the variable toplevel-score-handler at toplevel. (The default handler is defined in the file ../scm/lily-library.scm and set in the file ../ly/declarations-init.ly.)

- A \book block logically combines multiple movements (i.e., multiple \score blocks) in one document. If there are a number of \scores, one output file will be created for each \book block, in which all corresponding movements are concatenated. The only reason to explicitly specify \book blocks in a .ly file is if you wish to create multiple output files from a single input file. One exception is within lilypond-book documents, where you explicitly have to add a \book block if you want more than a single \score or \markup in the same example. This behavior can be changed by setting the variable toplevel-book-handler at toplevel. The default handler is defined in the init file ../scm/lily.scm.

- A \bookpart block. A book may be divided into several parts, using \bookpart blocks, in order to ease the page breaking, or to use different \paper settings in different parts.

- A compound music expression, such as

> { c'4 d' e'2 }

This will add the piece in a \score and format it in a single book together with all other toplevel \scores and music expressions. In other words, a file containing only the above music expression will be translated into

\begin{verbatim}
\book {
    \score {
        \new Staff {
            \new Voice {
                { c'4 d' e'2 }
            }
        }
        \layout { }
    }
    \paper { }
}\end{verbatim}
This behavior can be changed by setting the variable `toplevel-music-handler` at toplevel. The default handler is defined in the init file `../scm/lily.scm`.

- A markup text, a verse for example
  ```
  \markup {
    2. The first line verse two.
  }
  ```
  Markup texts are rendered above, between or below the scores or music expressions, wherever they appear.

- A variable, such as
  ```
  foo = { c4 d e d }
  ```
  This can be used later on in the file by entering `\foo`. The name of a variable should not contain (ASCII) numbers, multiple underscores, multiple dashes or space characters. All other characters Unicode provides are allowed, for example Latin, Greek, Chinese or Cyrillic. Non-adjacent single underscores and dashes are allowed, too. In other words, variable names like `HornIII` or `αβγXII` work.

Any combination of characters is allowed if the variable name is enclosed in double quotation marks. In this case backslashes and double quotation marks need to be escaped with backslashes (not that you actually should use them). Examples: "foo bar", "a-b-c", "Horn 3".

The following example shows three things that may be entered at toplevel
```
\layout {
  % Don't justify the output
  ragged-right = ##t
}

\header {
  title = "Do-re-mi"
}

{ c'4 d' e2 }
``` At any point in a file, any of the following lexical instructions can be entered:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- A single-line comment, introduced by a leading `%` sign.
- A multi-line comment delimited by `%%`.

Whitespace between items in the input stream is generally ignored, and may be freely omitted or extended to enhance readability. However, whitespace should always be used in the following circumstances to avoid errors:

- Around every opening and closing curly bracket.
- After every command or variable, i.e., every item that begins with a `\` sign.
- After every item that is to be interpreted as a Scheme expression, i.e., every item that begins with a `#` sign.
- To separate all elements of a Scheme expression.
• In **lyricmode** before and after `\set` and `\override` commands.

**See also**


Notation Reference: [Titles explained], page 492, Section 4.2.1 [The `\layout` block], page 557.

### 3.2 Titles and headers

Almost all printed music includes a title and the composer’s name; some pieces include a lot more information.

#### 3.2.1 Creating titles headers and footers

**Titles explained**

Each `\book` block in a single input file produces a separate output file, see Section 3.1.5 [File structure], page 490. Within each output file three types of titling areas are provided: **Book Titles** at the beginning of each book, **Bookpart Titles** at the beginning of each bookpart and **Score Titles** at the beginning of each score.

Values of titling fields such as `title` and `composer` are set in `\header` blocks. (For the syntax of `\header` blocks and a complete list of the fields available by default see [Default layout of bookpart and score titles], page 495). Book Titles, Bookpart Titles and Score Titles can all contain the same fields, although by default the fields in Score Titles are limited to `piece` and `opus`.

`\header` blocks may be placed in four different places to form a descending hierarchy of `\header` blocks:

- At the top of the input file, before all `\book`, `\bookpart`, and `\score` blocks.
- Within a `\book` block but outside all the `\bookpart` and `\score` blocks within that book.
- Within a `\bookpart` block but outside all `\score` blocks within that bookpart.
- Within a `\score` block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy, so:

- A Book Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block. The resulting fields are used to print the Book Title for that book, providing that there is other material which generates a page at the start of the book, before the first bookpart. A single `\pageBreak` will suffice.
- A Bookpart Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, and further modified by fields set in the `\bookpart` block. The resulting values are used to print the Bookpart Title for that bookpart.
- A Score Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, further modified by fields set in the `\bookpart` block and finally modified by fields set in the `\score` block. The resulting values are used to print the Score Title for that score. Note, though, that only `piece` and `opus` fields are printed by default in Score Titles unless the `\paper` variable, `print-all-headers`, is set to `#t`.

It is not necessary to provide `\header` blocks in all four places: any or even all of them may be omitted. Similarly, simple input files may omit the `\book` and `\bookpart` blocks, leaving them to be created implicitly.

If the book has only a single score, the `\header` block should normally be placed at the top of the file so that just a Bookpart Title is produced, making all the titling fields available for use.
If the book has multiple scores a number of different arrangements of \header blocks are possible, corresponding to the various types of musical publications. For example, if the publication contains several pieces by the same composer a \header block placed at the top of the file specifying the book title and the composer with \header blocks in each \score block specifying the piece and/or opus would be most suitable, as here:

\header{
  title = "SUITE I."
  composer = "J. S. Bach."
}

\score{
  \new Staff \relative {
    \clef bass
    \key g \major
    \repeat unfold 2 { g,16(d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16(e' c') b c e, c' e, } |
  }
  \header{
    piece = "Prélude."
  }
}

\score{
  \new Staff \relative {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b( c d e) fis( g a b) |
  }
  \header{
    piece = "Allemande."
  }
}

**SUITE I.**

J. S. Bach.

Prélude.

![Prélude](image1)

Allemande.

![Allemande](image2)
More complicated arrangements are possible. For example, text fields from the header block in a book can be displayed in all Score Titles, with some fields over-ridden and some manually suppressed:

\book{
  \paper{
    print-all-headers = ##t
  }
  \header{
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    \%
    % Do not display the default LilyPond footer for this book
    tagline = ##f
  }
  \markup{\vspace #1}
}
\score{
  \new PianoStaff<>
    \new Staff{ s1 }
    \new Staff{ \clef "bass" s1 }
  >>
  \header{
    title = "PRAELUDIUM I"
    opus = "BWV 846"
    \%
    % Do not display the subtitle for this score
    subtitle = ##f
  }
}
\score{
  \new PianoStaff<>
    \new Staff{ s1 }
    \new Staff{ \clef "bass" s1 }
  >>
  \header{
    title = "FUGA I"
    subsubtitle = "A 4 VOCI"
    opus = "BWV 846"
    \%
    % Do not display the subtitle for this score
    subtitle = ##f
  }
}
}
DAS WOHLTEMPERIRTE CLAVIER
TEIL I

PRAELUDIUM I

FUGA I
A 4 VOCI

See also
Notation Reference: Section 3.1.5 [File structure], page 490, [Default layout of bookpart and score titles], page 495, [Custom layout for titles], page 500.

Default layout of bookpart and score titles
This example demonstrates all printed \header variables:

\book {
  \header {
    % The following fields are centered
dedication = "Dedication"
title = "Title"
subtitle = "Subtitle"
subsubtitle = "Subsubtitle"
    % The following fields are evenly spread on one line
    % the field "instrument" also appears on following pages
instrument = \markup \with-color #green "Instrument"
poet = "Poet"
composer = "Composer"
    % The following fields are placed at opposite ends of the same line
meter = "Meter"
arranger = "Arranger"
    % The following fields are centered at the bottom
tagline = "The tagline goes at the bottom of the last page"
copyright = "The copyright goes at the bottom of the first page"
  }
}\score {
{ s1 }
\header {
  % The following fields are placed at opposite ends of the same line
  piece = "Piece 1"
  opus = "Opus 1"
}
\score {
  { s1 }
  \header {
    % The following fields are placed at opposite ends of the same line
    piece = "Piece 2 on the same page"
    opus = "Opus 2"
  }
}
\pageBreak
\score {
  { s1 }
  \header {
    % The following fields are placed at opposite ends of the same line
    piece = "Piece 3 on a new page"
    opus = "Opus 3"
  }
}

Dedication

Title

Subtitle

Subsubtitle

Poet

Meter

Instrument

Composer

Arranger

Opus 1

Piece 1

\timesig.C44/clefs.G

Piece 2 on the same page

\timesig.C44/clefs.G

The copyright goes at the bottom of the first page
The tagline goes at the bottom of the last page

Note that

- The instrument name will be repeated on every page.
- Only `piece` and `opus` are printed in a `score` when the paper variable `print-all-headers` is set to `##f` (the default).
- Text fields left unset in a `header` block are replaced with `null` markups so that the space is not wasted.
- The default settings for `scoreTitleMarkup` place the `piece` and `opus` text fields at opposite ends of the same line.

To change the default layout see [Custom layout for titles], page 500.

If a `book` block starts immediately with a `bookpart` block, no Book Title will be printed, as there is no page on which to print it. If a Book Title is required, begin the `book` block with some markup material or a `pageBreak` command.

Use the `breakbefore` variable inside a `header` block that is itself in a `score` block, to make the higher-level `header` block titles appear on the first page on their own, with the music (defined in the `score` block) starting on the next.

```
\book {
  \header {
    title = "This is my Title"
    subtitle = "This is my Subtitle"
    copyright = "This is the bottom of the first page"
  }
  \score {
    \repeat unfold 4 { c' ' c' ' c' ' c' ' }
    \header {
      piece = "This is the Music"
      breakbefore = ##t
    }
  }
}
```
**This is my Title**  
**This is my Subtitle**

This is the bottom of the first page

2  
This is the Music

\[ \begin{align*}  
\text{Music engraving by LilyPond 2.21.5—www.lilypond.org} 
\end{align*} \]

**See also**

Notation Reference: [Custom layout for titles], page 500, Section 3.1.5 [File structure], page 490.  
Installed Files: ly/titling-init.ly.

**Default layout of headers and footers**

*Headers* and *footers* are lines of text appearing at the top and bottom of pages, separate from the main text of a book. They are controlled by the following `\texttt{paper}` variables:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

These markup variables can only access text fields from top-level `\texttt{header}` blocks (which apply to all scores in the book) and are defined in `ly/titling-init.ly`. By default:

- page numbers are automatically placed on the top far left (if even) or top far right (if odd), starting from the second page.
• the **instrument** text field is placed in the center of every page, starting from the second page.
• the **copyright** text is centered on the bottom of the first page.
• the **tagline** is centered on the bottom of the last page, and below the copyright text if there is only a single page.

The default LilyPond footer text can be changed by adding a **tagline** in the top-level \header block.

\book {
  \header {
    tagline = "... music notation for Everyone"
  }
  \score {
    \relative {
      c'4 d e f
    }
  }
}

... music notation for Everyone

To remove the default LilyPond footer text, the tagline can be set to `##f`.

### 3.2.2 Custom titles, headers and footers

**Custom text formatting for titles**

Standard \markup commands can be used to customize any header, footer and title text within the \header block.

\score {
  { s1 }
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
}

**PRAELUDIUM I**  

`BWV 846`
See also

Notation Reference: Section 1.8.2 (Formatting text), page 252.

Custom layout for titles

\markup commands in the \header block are useful for simple text formatting, but they do not allow precise control over the placement of titles. To customize the placement of the text fields, change either or both of the following \paper variables:

- bookTitleMarkup
- scoreTitleMarkup

The placement of titles when using the default values of these \markup variables is shown in the examples in [Default layout of bookpart and score titles], page 495.

The default settings for scoreTitleMarkup as defined in ly/titling-init.ly are:

```
scoreTitleMarkup = \markup { \column { \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 } \fill-line { \fromproperty #'header:piece \fromproperty #'header:opus } } }
```

This places the piece and opus text fields at opposite ends of the same line:

```
\score { \s1 } \header { piece = "PRAELUDIUM I" opus = "BWV 846" }
```

PRAELUDIUM I     BWV 846

This example redefines scoreTitleMarkup so that the piece text field is centered and in a large, bold font.

```
\book { \paper { indent = 0\mm scoreTitleMarkup = \markup { \fill-line { \null \fontsize #4 \bold \fromproperty #'header:piece \fromproperty #'header:opus } } } \header { tagline = ##f } }
```
Text fields not normally effective in score \header blocks can be printed in the Score Title area if print-all-headers is placed inside the \paper block. A disadvantage of using this method is that text fields that are intended specifically for the Bookpart Title area need to be manually suppressed in every \score block. See [Titles explained], page 492.

To avoid this, add the desired text field to the scoreTitleMarkup definition. In the following example, the composer text field (normally associated with bookTitleMarkup) is added to scoreTitleMarkup, allowing each score to list a different composer:

\book {
\paper {
indent = 0\mm
scoreTitleMarkup = \markup {
fill-line {
null
\fontsize #4 \bold \fromproperty #'header:piece
\fromproperty #'header:composer
}
}
}
\header { tagline = ##f }
\score {
{ s1 }
\header {
piece = "MENUET"
composer = "Christian Petzold"
}
}
\score {
{ s1 }
\header {
piece = "RONDEAU"
composer = "François Couperin"
}
}
}
It is also possible to create your own custom text fields, and refer to them in the markup definition.

\begin{verbatim}
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #'(direction . ,UP)
        \dir-column {
          \center-align \fontsize #-1 \bold
          \fromproperty #'header:mycustomtext \textcopyright User-defined field
          \center-align \fontsize #4 \bold
          \fromproperty #'header:piece
        }
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \set s1 {
      piece = "FUGA I"
      mycustomtext = "A 4 VOCI" \textcopyright User-defined field
      opus = "BWV 846"
    }
  }
}
\end{verbatim}

See also
Notation Reference: [Titles explained], page 492.
Custom layout for headers and footers

`\markup` commands in the `\header` block are useful for simple text formatting, but they do not allow precise control over the placement of headers and footers. To customize the placement of the text fields, use either or both of the following `\paper` variables:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

The `\markup` command `\on-the-fly` can be used to add markup conditionally to header and footer text defined within the `\paper` block, using the following syntax:

```latex
variable = \markup {
  \on-the-fly \procedure markup
  ...
}
```

The `procedure` is called each time the `\markup` command in which it appears is evaluated. The `procedure` should test for a particular condition and interpret (i.e., print) the `markup` argument if and only if the condition is true.

A number of ready-made procedures for testing various conditions are provided:

<table>
<thead>
<tr>
<th>Procedure name</th>
<th>Condition tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>print-page-number-check-first</td>
<td>should this page number be printed?</td>
</tr>
<tr>
<td>create-page-number-stencil</td>
<td>print-page-numbers true?</td>
</tr>
<tr>
<td>print-all-headers</td>
<td>print-all-headers true?</td>
</tr>
<tr>
<td>first-page</td>
<td>first page in the book?</td>
</tr>
<tr>
<td>not-first-page</td>
<td>not first page in the book?</td>
</tr>
<tr>
<td>(on-page nmbr)</td>
<td>page number = nmbr?</td>
</tr>
<tr>
<td>last-page</td>
<td>last page in the book?</td>
</tr>
<tr>
<td>part-first-page</td>
<td>first page in the book part?</td>
</tr>
<tr>
<td>not-part-first-page</td>
<td>not first page in the book part?</td>
</tr>
<tr>
<td>part-last-page</td>
<td>last page in the book part?</td>
</tr>
<tr>
<td>not-single-page</td>
<td>pages in book part &gt; 1?</td>
</tr>
</tbody>
</table>

The following example centers page numbers at the bottom of every page. First, the default settings for `oddHeaderMarkup` and `evenHeaderMarkup` are removed by defining each as a `null` markup. Then, `oddFooterMarkup` is redefined with the page number centered. Finally, `evenFooterMarkup` is given the same layout by defining it as `\oddFooterMarkup`:

```latex
\book {
  \paper {
    print-page-number = ##t
    print-first-page-number = ##t
    oddHeaderMarkup = \markup \null
    evenHeaderMarkup = \markup \null
    oddFooterMarkup = \markup {
      \fill-line {
        \on-the-fly \print-page-number-check-first
        \fromproperty # \text:'page:page-number-string}
    }
  }
}
```
Several \on-the-fly conditions can be combined with an ‘and’ operation, for example,
\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }
determines if the output is a single page.

See also
Notation Reference: [Titles explained], page 492, [Default layout of bookpart and score titles], page 495.
Installed Files: ../ly/titling-init.ly.

3.2.3 Creating output file metadata
In addition to being shown in the printed output, \header variables are also used to set metadata for output files. For example, with PDF files, this metadata could be displayed by PDF readers as the properties of the PDF file. For each type of output file, only the \header definitions of blocks that define separate files of that type, and blocks higher in the block hierarchy, will be consulted. Therefore, for PDF files, only the \book level and the top level \header definitions affect the document-wide PDF metadata, whereas for MIDI files, all headers above or at the \score level are used.

For example, setting the title property of the header block to ‘Symphony I’ will also give this title to the PDF document, and use it as the sequence name of the MIDI file.
\header {
  title = "Symphony I"
}
If you want to set the title of the printed output to one value, but have the title property of
the PDF to have a different value, you can use \pdftitle, as below.

```latex
\header {
  title = "Symphony I"
  pdftitle = "Symphony I by Beethoven"
}
```

The variables \texttt{title}, \texttt{subject}, \texttt{keywords}, \texttt{subtitle}, \texttt{composer}, \texttt{arranger}, \texttt{poet}, \texttt{author} and \texttt{copyright} all set PDF properties and can all be prefixed with 'pdf' to set a PDF property to
a value different from the printed output.

The PDF property \texttt{Creator} is automatically set to ‘LilyPond’ plus the current LilyPond
version, and \texttt{CreationDate} and \texttt{ModDate} are both set to the current date and time. \texttt{ModDate}
can be overridden by setting the header variable \texttt{moddate} (or \texttt{pdfmoddate}) to a valid PDF date
string.

The \texttt{title} variable sets also the sequence name for MIDI. The \texttt{midititle} variable can be
used to set the sequence name independently of the value used for typeset output.

### 3.2.4 Creating footnotes

Footnotes may be used in many different situations. In all cases, a ‘footnote mark’ is placed
as a reference in text or music, and the corresponding ‘footnote text’ appears at the bottom
of the same page, separated from the music by a horizontal line. The appearance of this separator
can be changed by setting the paper variable \texttt{footnote-separator-markup}, see [Miscellaneous
paper variables], page 555.

Footnotes within music expressions and footnotes in stand-alone text outside music expres-
sions are created in different ways.

### Footnotes in music expressions

**Music footnotes overview**

Footnotes in music expressions fall into two categories:

- **Event-based footnotes**
  are attached to a particular event. Examples for such events are single notes, artic-
ulations (like fingerling indications, accents, dynamics), and post-events (like slurs
and manual beams). The general form for event-based footnotes is as follows:

  \[ [\text{direction}] \\text{\footnote} [\text{mark}] \text{offset footnote music} \]

- **Time-based footnotes**
  are bound to a particular point of time in a musical context. Some commands like
  \texttt{\time} and \texttt{\clef} don't actually use events for creating objects like time signatures
  and clefs. Neither does a chord create an event of its own: its stem or flag is created
  at the end of a time step (nominally through one of the note events inside). Exactly
  which of a chord’s multiple note events will be deemed the root cause of a stem or
  flag is undefined. So for annotating those, time-based footnotes are preferable as
  well.

  A time-based footnote allows such layout objects to be annotated without referring
to an event. The general form for Time-based footnotes is:

  \texttt{\footnote [\text{mark}] \text{offset footnote} [\text{Context}.\text{GrobName}]}

  The elements for both forms are:

  - **direction** If (and only if) the \texttt{\footnote} is being applied to a post-event or articulation, it
    must be preceded with a direction indicator (\texttt{-}, \texttt{.}, \texttt{^}) in order to attach \texttt{music}
    (with a footnote mark) to the preceding note or rest.
**mark** is a markup or string specifying the footnote mark which is used for marking both the reference point and the footnote itself at the bottom of the page. It may be omitted (or equivalently replaced with `\default`) in which case a number in sequence will be generated automatically. Such numerical sequences restart on each page containing a footnote.

**offset** is a number pair such as `#(2 . 1)` specifying the X and Y offsets in units of staff-spaces from the boundary of the object where the mark should be placed. Positive values of the offsets are taken from the right/top edge, negative values from the left/bottom edge and zero implies the mark is centered on the edge.

**Context** is the context in which the grob being footnoted is created. It may be omitted if the grob is in a bottom context, e.g., a **Voice** context.

**GrobName** specifies a type of grob to mark (like `Flag`). If it is specified, the footnote is not attached to a music expression in particular, but rather to all grobs of the type specified which occur at that moment of musical time.

**footnote** is the markup or string specifying the footnote text to use at the bottom of the page.

**music** is the music event or post-event or articulation that is being annotated.

**Event-based footnotes**
A footnote may be attached to a layout object directly caused by the event corresponding to **music** with the syntax:

```
\footnote [mark] offset footnote music
```

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "A note" a4
    a4
    \footnote #'(2 . 2) "A rest" r4
    a4
  }
}
```

```
\note\^1\-\ rests.2/\noteheads.s2\timesens.C44

\note\^1\-\ A note
\note\^2\-\ A rest
```

Marking a **whole** chord with an event-based footnote is not possible: a chord, even one containing just a single note, does not produce an actual event of its own. However, individual notes **inside** of the chord can be marked:

```
\book {
```
If the footnote is to be attached to a post-event or articulation the \footnote command must be preceded by a direction indicator, -, _, ^, and followed by the post-event or articulation to be annotated as the music argument. In this form the \footnote can be considered to be simply a copy of its last argument with a footnote mark attached to it. The syntax is:

direction \footnote [mark] offset footnote music

\book {
  \header { tagline = ##f }
  \relative c' {
    a'4_\footnote #'(0 . -1) "A slur forced down" ( 
    b8^\footnote #'(1 . 0.5) "A manual beam forced up" [ 
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
}

A slur forced down
\footnote{A manual beam forced up

\footnote{Tenuto

\book {

Time-based footnotes

If the layout object being footmarked is indirectly caused by an event (like an Accidental or Stem caused by a NoteHead event), the GrobName of the layout object is required after the footnote text instead of music:

\book {

\footnote{A slur forced down

\footnote{A manual beam forced up

\footnote{Tenuto
Chapter 3: General input and output

Note, however, that when a GrobName is specified, a footnote will be attached to all grobs of that type at the current time step:

\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "A flat" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Articulation" Script
  }
}

\header { tagline = ##f }
\relative c' {
  \footnote #'(-1 . 3) "A flat" Accidental
  \footnote #'(-1 . 0.5) "Another flat" Accidental
  \footnote '(1 . -2) "A stem" Stem
}

\header { tagline = ##f }
\relative c' {
  \footnote #'(-1 . 3) "A flat" Accidental
  <ees ges bes>4
  \footnote #'(2 . 0.5) "Articulation" Script
  c'->-5.
}

A note inside of a chord can be given an individual (event-based) footnote. A `NoteHead` is the only grob directly caused from a chord note, so an event-based footnote command is only suitable for adding a footnote to the `NoteHead` within a chord. All other chord note grobs are indirectly caused. The \footnote command itself offers no syntax for specifying both a
particular grob type as well as a particular event to attach to. However, one can use a time-based footnote command for specifying the grob type, and then prefix this command with single in order to have it applied to just the following event:

\book {
  \header { tagline = ##f }
  \relative c' {
    < \footnote #'(1 . -2) "An A" a
    \single \footnote #(-1 . -1) "A sharp" Accidental
    cis
    \single \footnote #'(0.5 . 0.5) "A flat" Accidental
    ees fis
    >2
  }
}

\begin{musicfigure}
\begin{musicstaff}
\relative c' {
  r1 |
  \footnote #(-0.5 . -1) "Meter change" Staff.TimeSignature
  \time 3/4
  \footnote #'(1 . -1) "Chord stem" Stem
  <c e g>4 q q
  \footnote #(-0.5 . 1) "Bar line" Staff.BarLine
}\end{musicstaff}
\end{musicfigure}

1A flat
2A sharp
3An A

Note: When footnotes are attached to several musical elements at the same musical moment, as they are in the example above, the footnotes are numbered from the higher to the lower elements as they appear in the printed output, not in the order in which they are written in the input stream.

Layout objects like clefs and key-change signatures are mostly caused as a consequence of changed properties rather than actual events. Others, like bar lines and bar numbers, are a direct consequence of timing. For this reason, footnotes on such objects have to be based on their musical timing. Time-based footnotes are also preferable when marking features like stems and beams on chords: while such per-chord features are nominally assigned to one event inside the chord, relying on a particular choice would be imprudent.

The layout object in question must always be explicitly specified for time-based footnotes, and the appropriate context must be specified if the grob is created in a context other than the bottom context.
Custom marks can be used as alternatives to numerical marks, and the annotation line joining the marked object to the mark can be suppressed:

\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "* The first note" } a'4 b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
    \markup { \super "$" \italic " The second note" } e c4
    \once \override Score.FootnoteItem.annotation-line = ##f
    \footnote \markup \tiny "+" #'(0.1 . 0.1)
    \markup { \super "+" \italic " Editorial" } \p
  }
}

* The first note
$ The second note
+ Editorial

More examples of custom marks are shown in [Footnotes in stand-alone text], page 510.

Footnotes in stand-alone text
These are for use in markup outside of music expressions. They do not have a line drawn to their point of reference: their marks simply follow the referenced markup. Marks can be inserted
automatically, in which case they are numerical. Alternatively, custom marks can be provided manually.

Footnotes to stand-alone text with automatic and custom marks are created in different ways.

**Footnotes in stand-alone text with automatic marks**

The syntax of a footnote in stand-alone text with automatic marks is

\markup { ... \footnote text footnote ... }

The elements are:

- **text**: is the markup or string to be marked.
- **footnote**: is the markup or string specifying the footnote text to use at the bottom of the page.

For example:

\book {
  \header { tagline = ##f }
  \markup {
    "A simple" \footnote "tune" \italic "By me"
    "is shown below. It is a"
    \footnote "recent" \italic "Aug 2012"
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}

A simple tune is shown below. It is a recent composition.

\begin{music}
\relative { c'4 d e f g a'16}\end{music}

---

*By me*

*Aug 2012*

**Footnotes in stand-alone text with custom marks**

The syntax of a footnote in stand-alone text with custom marks is

\markup { ... \footnote mark footnote ... }

The elements are:

- **mark**: is a markup or string specifying the footnote mark which is used for marking the reference point. Note that this mark is *not* inserted automatically before the footnote itself.
footnote is the markup or string specifying the footnote text to use at the bottom of the page, preceded by the mark.

Any easy-to-type character such as * or + may be used as a mark, as shown in [Footnotes in music expressions], page 505. Alternatively, ASCII aliases may be used (see [ASCII aliases], page 526):

\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote "*" italic "* By me"
    "is shown below. It is a recent"
    \footnote super &dagger; \concat {
      \super &dagger; italic " Aug 2012"
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}

A simple tune * is shown below. It is a recent † composition.

* By me
† Aug 2012

Unicode character codes may also be used to specify marks (see [Unicode], page 525):

\book {
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote super \char##x00a7 \concat {
      \super \char##x00a7 italic " By me"
    }
    "is shown below. It is a recent"
    \footnote super \char##x00b6 \concat {
      \super \char##x00b6 italic " Aug 2012"
    }
    "composition."
}
A simple tune is shown below. It is a recent composition.

\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
  c'1
  \pageBreak \mark A \label #'markA
  \}
}

§ By me
¶ Aug 2012

See also


Notation Reference: [ASCII aliases], page 526, [Balloon help], page 237, Section A.13 [List of special characters], page 767, [Text marks], page 248, [Text scripts], page 245, [Unicode], page 525.

Internals Reference: Section “FootnoteEvent” in Internals Reference, Section “FootnoteItem” in Internals Reference, Section “FootnoteSpanner” in Internals Reference, Section “FootnoteEngraver” in Internals Reference.

Known issues and warnings

Multiple footnotes for the same page can only be stacked, one above the other; they cannot be printed on the same line.

Footnotes cannot be attached to MultiMeasureRests or automatic beams or lyrics.

Footnote marks may collide with staves, \markup objects, other footnote marks and annotation lines.

3.2.5 Reference to page numbers

A particular place of a score can be marked using the \label command, either at top-level or inside music. This label can then be referred to in a markup, to get the number of the page where the marked point is placed, using the \page-ref markup command.

\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    c'1
    \pageBreak \mark A \label #'markA
The first score begins on page 1
Mark A is on page 2

The \page-ref markup command takes three arguments:
1. the label, a Scheme symbol, for example #'firstScore;
2. a markup that will be used as a gauge to estimate the dimensions of the markup;
3. a markup that will be used in place of the page number if the label is not known.

The reason why a gauge is needed is that, at the time markups are interpreted, the page breaking has not yet occurred, so the page numbers are not yet known. To work around this issue, the actual markup interpretation is delayed to a later time; however, the dimensions of the markup have to be known before, so a gauge is used to decide these dimensions. If the book has between 10 and 99 pages, it may be "00", ie. a two digit number.

Predefined commands
\label, \page-ref.

3.2.6 Table of contents
A table of contents is included using the \markuplist \table-of-contents command. The elements which should appear in the table of contents are entered with the \tocItem command, which may be used either at top-level, or inside a music expression.

\markuplist \table-of-contents
\pageBreak

\tocItem \markup "First score"
\score {
  { c'4 % ... 
    \tocItem \markup "Some particular point in the first score"
    d'4 % ...
  }
}
\tocItem \markup "Second score"
\score {
  { 
e'4 % ... 
  \tocItem actI \markup "Act I"
f'4 % ... 
  \tocItem actI.sceneI \markup "Scene 1"
g'4 % ... 
  \tocItem actI.sceneI.recitativo \markup "Recit."
a'4 % ... 
}
}

Optionally, a label can be associated with a particular item, or a hierarchical list of existing
labels, finishing with that item’s label. That latter case allows to mark the item as a ‘child’ of
the preceeding labeled items, thus making the score’s structure apparent in the table of contents.

Markups used for formatting the table of contents are defined in the \paper block. There
are three ‘pre-defined’ markups already available:

- \tocTitleMarkup
  Used for formatting the title of the table of contents.
  \tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table of Contents" \null }
  }

- \tocItemMarkup
  Used for formatting the elements within the table of contents.
  \tocItemMarkup = \markup \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }

- \tocFormatMarkup
  How the table’s top level entries will be formatted (if there are several hierarchical levels).
  This is actually a procedure, as explained in Section “Markup construction in Scheme” in Extending.
  \tocFormatMarkup = #make-bold-markup

- \tocIndentMarkup
  Used to define how the outline’s hierarchy will be made apparent. This markup is printed
  zero, one or several times depending on the level of each entry.
  \tocIndentMarkup = \markup \hspace #4

Any of these variables can be changed.

Here is an example translating the table of contents’ title into French:
\paper {
  \tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
}

Here is an example changing the font-size of the elements in the table of contents:
\tocItemMarkup = \markup \large \fill-line {
Note how the element text and page numbers are referred to in the tocItemMarkup definition.

The \tocItemWithDotsMarkup command can be included within the tocItemMarkup to fill the line, between a table of contents item and its corresponding page number, with dots:

\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}
\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}

Table of Contents

Allegro .................................................. 1
Largo .................................................... 1

In addition to the built-in outline mechanism, custom commands can also be defined to build a more personalized table of contents with different markups. In the following example, a new style is defined for entering acts and scenes in the table of contents of an opera:

A new markup variable (called tocActMarkup) is defined in the \paper block:

\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}

A custom music function (tocAct) is then created – which uses the new tocActMarkup markup definition, and allows to specify a label for each act.

tocAct =
\( \#(\text{define-music-function \( \text{label text} \) \text{symbol? \( \text{markup?} \)}}\)
\( \text{(add-to-c-item! 'tocActMarkup label text)} \)\)

Using these custom definitions and modifying some of the existing definitions, the source file could then be written as follows:
Chapter 3: General input and output

517

Table of Contents
Atto Primo
Coro. Viva il nostro Alcide . . . . . . . . . .
Cesare. Presti omai l'Egizia terra
. . . . . .

1
1
1

Atto Secondo
Sinfonia
. . . . . . . . . . . . . . . . . .
Cleopatra. V'adoro, pupille, saette d'Amore . .

1
1

The previous example also demonstrates how to use the \fill-with-pattern markup command
within the context of a table of contents.

See also
Installed Files: ly/toc-init.ly.

Predefined commands
\table-of-contents,

\tocItem,

tocItemMarkup,

tocTitleMarkup,

tocFormatMarkup,

tocIndentMarkup.

3.3 Working with input files
3.3.1 Including LilyPond files
A large project may be split up into separate files. To refer to another file, use
\include "otherfile.ly"
The line \include "otherfile.ly" is equivalent to pasting the contents of otherfile.ly
into the current file at the place where the \include appears. For example, in a large project
you might write separate files for each instrument part and create a “full score” file which
brings together the individual instrument files. Normally the included file will define a number
of variables which then become available for use in the full score file. Tagged sections can
be marked in included files to assist in making them usable in different places in a score, see
Section 3.3.2 [Different editions from one source], page 519.
Files in the current working directory may be referenced by specifying just the file name
after the \include command. Files in other locations may be included by giving either a full
path reference or a relative path reference (but use the UNIX forward slash, /, rather than the
DOS/Windows back slash, \, as the directory separator.) For example, if stuff.ly is located
one directory higher than the current working directory, use
\include "../stuff.ly"
or if the included orchestral parts files are all located in a subdirectory called parts within the
current directory, use
\include "parts/VI.ly"
\include "parts/VII.ly"
... etc
Files which are to be included can also contain \include statements of their own. These
second-level \include statements are then interpreted relatively to the path of the file containing
that command, which is convenient for multiple files located in the same subdirectory. For


example, a general library, libA, may itself use sub-files which are `\include`d by the entry file of that library, like this:

```
libA/
   libA.ly
   A1.ly
   A2.ly
   ...
```

then the entry file, `libA.ly`, will contain

```
\include "A1.ly"
\include "A2.ly"
...
```

Any `.ly` file can then include the entire library simply with

```
\include "~/libA/libA.ly"
```

However, that behavior can be changed globally by passing the option `-drelative-includes=#f` option at the command line (or by adding `#(ly:set-option 'relative-includes #f)` at the top of the main input file). In that case, each file will be included relatively to the location of the main file, regardless of where its `\include` statement is located. Complex file structures, that require to `\include` both files relative to the main directory and files relative to some other directory, may even be devised by setting `relative-includes` to `#f` or `#t` at appropriate places in the files.

Files can also be included from a directory in a search path specified as an option when invoking LilyPond from the command line. The included files are then specified using just their file name. For example, to compile `main.ly` which includes files located in a subdirectory called `parts` by this method, cd to the directory containing `main.ly` and enter

```
lilypond --include=parts main.ly
```

and in `main.ly` write

```
\include "VI.ly"
\include "VII.ly"
...
```

Files which are to be included in many scores may be placed in the LilyPond directory `../ly`. (The location of this directory is installation-dependent - see Section “Other sources of information” in Learning Manual). These files can then be included simply by naming them on an `\include` statement. This is how the language-dependent files like `english.ly` are included.

LilyPond includes a number of files by default when you start the program. These includes are not apparent to the user, but the files may be identified by running `lilypond --verbose` from the command line. This will display a list of paths and files that LilyPond uses, along with much other information. Alternatively, the more important of these files are discussed in Section “Other sources of information” in Learning Manual. These files may be edited, but changes to them will be lost on installing a new version of LilyPond.

Some simple examples of using `\include` are shown in Section “Scores and parts” in Learning Manual.

**See also**


**Known issues and warnings**

If an included file is given a name which is the same as one in LilyPond’s installation files, LilyPond’s file from the installation files takes precedence.
3.3.2 Different editions from one source

Several methods can be used to generate different versions of a score from the same music source. Variables are perhaps the most useful for combining lengthy sections of music and/or annotation. Tags are more useful for selecting one section from several alternative shorter sections of music, and can also be used for splicing pieces of music together at different points.

Whichever method is used, separating the notation from the structure of the score will make it easier to change the structure while leaving the notation untouched.

Using variables

If sections of the music are defined in variables they can be reused in different parts of the score, see Section “Organizing pieces with variables” in Learning Manual. For example, an a cappella vocal score frequently includes a piano reduction of the parts for rehearsal purposes which is identical to the vocal music, so the music need be entered only once. Music from two variables may be combined on one staff, see [Automatic part combining], page 183. Here is an example:

```latex
sopranoMusic = \relative { a'4 b c b8( a) }
altoMusic = \relative { e'4 e e f } 
tenorMusic = \relative { c'4 b e d8( c) }
bassMusic = \relative { a4 gis a d, }
allLyrics = \lyricmode { King of glo -- ry }
<<
\new Staff = "Soprano" \sopranoMusic
\new Lyrics \allLyrics
\new Staff = "Alto" \altoMusic
\new Lyrics \allLyrics
\new Staff = "Tenor" {
  \clef "treble_8"
  \tenorMusic }
\new Lyrics \allLyrics
\new Staff = "Bass" {
  \clef "bass"
  \bassMusic }
\new Lyrics \allLyrics
\new PianoStaff <<
  \new Staff = "RH" {
    \partCombine \sopranoMusic \altoMusic
  }
\new Staff = "LH" {
  \clef "bass"
  \partCombine \tenorMusic \bassMusic
}
>>
>>
```
Separate scores showing just the vocal parts or just the piano part can be produced by changing just the structural statements, leaving the musical notation unchanged.

For lengthy scores, the variable definitions may be placed in separate files which are then included, see Section 3.3.1 [Including LilyPond files], page 517.

Using tags

The \tag `partA` command marks a music expression with the name `partA`. Expressions tagged in this way can be selected or filtered out by name later, using either \keepWithTag #`name` or \removeWithTag #`name`. The result of applying these filters to tagged music is as follows:

**Filter**

- Tagged music preceded by \keepWithTag #`name` or \keepWithTag #`(name1 name2...)`
- Tagged music preceded by \removeWithTag #`name` or \removeWithTag #`(name1 name2...)`
- Tagged music not preceded by either \keepWithTag or \removeWithTag

**Result**

- Untagged music and music tagged with any of the given tag names is included; music tagged with any other tag name is excluded.
- Untagged music and music not tagged with any of the given tag names is included; music tagged with any of the given tag names is excluded.
- All tagged and untagged music is included.

The arguments of the \tag, \keepWithTag and \removeWithTag commands should be a symbol or list of symbols (such as #`score` or #`(violinI violinII)`, followed by a music expression. If and only if the symbols are valid LilyPond identifiers (alphabetic characters only, no numbers, underscores, or dashes) which cannot be confused with notes, the #` may be omitted and, as a shorthand, a list of symbols can use the dot separator: i.e., \tag #`(violinI violinII) can be written \tag violinI.violinII. The same applies to \keepWithTag and \removeWithTag. Tagging commands are music functions, thus they cannot be used to filter items that are not music expressions, such as \book or \score blocks.

In the following example, we see two versions of a piece of music, one showing trills with the usual notation, and one with trills explicitly expanded:
Alternatively, it is sometimes easier to exclude sections of music:

```latex
music = \relative {
    g'8. c32 d
    \tag #'trills { d8.\trill}
    \tag #'expand { \repeat unfold 3 { e32 d } }
    c32 d
}

\score {
    \keepWithTag #'trills \music
}
\score {
    \keepWithTag #'expand \music
}
```

Tagged filtering can be applied to articulations, texts, etc., by prepending
\texttt{-\tag #'your-tag}
to an articulation. For example, this would define a note with a conditional fingering indication and a note with a conditional annotation:

\c1\-\tag \#'finger \^4
\c1\-\tag \#'warn \"Watch!\"

Multiple tags may be placed on expressions with multiple \tag entries, or by combining multiple tags into one symbol list:

\music = \relative c'' {
  \tag \#'a \tag \#'both \{ a4 a a a \}
  \tag \#'(b both) \{ b4 b b b \}
}
<<
\keepWithTag \#'a \music
\keepWithTag \#'b \music
\keepWithTag \#'both \music
>>

Multiple \removeWithTag filters may be applied to a single music expression to remove several differently named tagged sections. Alternatively, you can use a single \removeWithTag with a list of tags.

\music = \relative c'' {
  \tag \#'A \{ a4 a a a \}
  \tag \#'B \{ b4 b b b \}
  \tag \#'C \{ c4 c c c \}
  \tag \#'D \{ d4 d d d \}
}
\new Voice {
  \removeWithTag \#'B
  \removeWithTag \#'C
  \music
  \removeWithTag \#'(B C)
  \music
}

Using two or more \keepWithTag filters on a single music expression will cause all of the tagged sections to be removed. The first filter will remove all except the one named and any
subsequent filters will remove the rest. Using one \keepWithTag command with a list of multiple
tags will only remove tagged sections that are not specified in that list.

\begin{verbatim}
music = \relative c' { 
\tag #'violinI { a4 a a a }
\tag #'violinII { b4 b b b }
\tag #'viola { c4 c c c }
\tag #'cello { d4 d d d }
}
\new Staff {
\keepWithTag #'(violinI violinII)
music
}
\end{verbatim}

will print \tags violinI and violinII but not viola or cello.

While \keepWithTag is convenient when dealing with one set of alternatives, the removal of
music tagged with unrelated tags is problematic when using them for more than one purpose.
In that case ‘groups’ of tags can be declared:

\begin{verbatim}
\tagGroup #'(violinI violinII viola cello)
\end{verbatim}

Now all the different tags belong to a single ‘tag group’. Note that individual tags cannot be
members of more than one tag group.

\begin{verbatim}
\keepWithTag #'violinI ...
\end{verbatim}

will now only show music tagged from violinI’s tag group and any music tagged with one of
the other tags will removed.

\begin{verbatim}
music = \relative { 
\tagGroup #'(violinI violinII viola cello)
\tag #'violinI { c'4"violinI" c c c }
\tag #'violinII { a2 a }
\tag #'viola { e8 e e2. }
\tag #'cello { d'2 d4 d }
R1""untagged"
}
\new Voice {
\keepWithTag #'violinI
music
}
\end{verbatim}

When using the \keepWithTag command, only tags from the tag groups of the tags given in
the command are visible.

Sometimes you want to splice some music at a particular place in an existing music expres-
sion. You can use \pushToTag and \appendToTag for adding material at the front or end of
the elements of an existing music construct. Not every music construct has elements, but sequential and simultaneous music are safe bets:

\[
\text{music} = \{ \text{tag } \text{"here}\{ \text{tag } \text{"here }<<c'>> } \} \}
\]

\[
\{ \text{pushToTag } \text{"here }c' \}
\text{pushToTag } \text{"here e'} \text{pushToTag } \text{"here }g' \text{music}
\text{appendToTag } \text{"here }c' \text{appendToTag } \text{"here e'} \text{appendToTag } \text{"here }g' \text{music}
\}
\]

Both commands get a tag, the material to splice in at every occurrence of the tag, and the tagged expression.

See also

Notation Reference: [Automatic part combining], page 183, Section 3.3.1 [Including LilyPond files], page 517.

Known issues and warnings
Calling \relative on a music expression obtained by filtering music through \keepWithTag or \removeWithTag might cause the octave relations to change, as only the pitches actually remaining in the filtered expression will be considered. Applying \relative first, before \keepWithTag or \removeWithTag, avoids this danger as \relative then acts on all the pitches as-input.

Using global settings
Global settings can be included from a separate file:

```
lilypond -dinclude-settings=MY_SETTINGS.ly MY_SCORE.ly
```

Groups of settings such as page size, font or type face can be stored in separate files. This allows different editions from the same score as well as standard settings to be applied to many scores, simply by specifying the proper settings file.

This technique also works well with the use of style sheets, as discussed in Section “Style sheets” in Learning Manual.

See also

Notation Reference: Section 3.3.1 [Including LilyPond files], page 517.

3.3.3 Special characters
Text encoding

LilyPond uses the character repertoire defined by the Unicode consortium and ISO/IEC 10646. This defines a unique name and code point for the character sets used in virtually all modern languages and many others too. Unicode can be implemented using several different encodings. LilyPond uses the UTF-8 encoding (UTF stands for Unicode Transformation Format) which represents all common Latin characters in one byte, and represents other characters using a variable length format of up to four bytes.

The actual appearance of the characters is determined by the glyphs defined in the particular fonts available - a font defines the mapping of a subset of the Unicode code points to glyphs. LilyPond uses the Pango library to layout and render multi-lingual texts.

LilyPond does not perform any input-encoding conversions. This means that any text, be it title, lyric text, or musical instruction containing non-ASCII characters, must be encoded in UTF-8. The easiest way to enter such text is by using a Unicode-aware editor and saving the file with UTF-8 encoding. Most popular modern editors have UTF-8 support, for example, vim, Emacs, jEdit, and Gedit do. All MS Windows systems later than NT use Unicode as their native character encoding, so even Notepad can edit and save a file in UTF-8 format. A more functional alternative for Windows is BabelPad.

If a LilyPond input file containing a non-ASCII character is not saved in UTF-8 format the error message

```lilypond
FT_Get_Glyph_Name () error: invalid argument
```

will be generated.

Here is an example showing Cyrillic, Hebrew and Portuguese text:

```
\relative {c''1 \mark \markup { \char \#x03EE }

\score { c' \mark \markup { \char \#x03BE }
```

Unicode

To enter a single character for which the Unicode code point is known but which is not available in the editor being used, use either `\char \#xhhhh` or `\char \#dddd` within a `\markup` block, where `hhhh` is the hexadecimal code for the character required and `dddd` is the corresponding decimal value. Leading zeroes may be omitted, but it is usual to specify all four characters in the hexadecimal representation. (Note that the UTF-8 encoding of the code point should not be used after `\char`, as UTF-8 encodings contain extra bits indicating the number of octets.) Unicode code charts and a character name index giving the code point in hexadecimal for any character can be found on the Unicode Consortium website, [http://www.unicode.org/](http://www.unicode.org/).

For example, `\char \#x03BE` and `\char \#958` would both enter the Unicode U+03BE character, which has the Unicode name “Greek Small Letter Xi”.

Any Unicode code point may be entered in this way and if all special characters are entered in this format it is not necessary to save the input file in UTF-8 format. Of course, a font containing all such encoded characters must be installed and available to LilyPond.

The following example shows Unicode hexadecimal values being entered in four places – in a rehearsal mark, as articulation text, in lyrics and as stand-alone text below the score:

```
\score { 
  \relative { 
    c''1 \mark \markup { \char \#x03EE }
```
Copyright 2008--2020 ©

To enter the copyright sign in the copyright notice use:

\header {
copyright = \markup { \char ##x00A9 "2008" }
}

ASCII aliases
A list of ASCII aliases for special characters can be included:

\paper {
#(include-special-characters)
}

\markup "&flqq; &ndash; &OE;uvre incompl` ete… &frqq;"

\score {
\new Staff { \repeat unfold 9 a'4 }
\addlyrics {
This is al -- so wor -- kin'"in ly -- rics: &ndash; &OE; &hellip;
}
}

\markup \column {
"The replacement can be disabled:" 
"&ndash; &OE; &hellip;"
\override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}

« – Œuvre incomplète... »

This is al-so workin’ in lyrics: – Œ...

The replacement can be disabled:
– Œ...
&ndash; &OE; &hellip;
You can also make your own aliases, either globally:

\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}

\markup "A 100 dpi."

A hundred dots per inch.

or locally:

\markup \replace #("100" . "hundred")
  ("dpi" . "dots per inch") "A 100 dpi."

A hundred dots per inch.

See also

Notation Reference: Section A.13 [List of special characters], page 767.

Installed Files: ly/text-replacements.ly.

3.4 Controlling output

3.4.1 Extracting fragments of music

It is possible to output one or more fragments of a score by defining the explicit location of
the music to be extracted within the layout block of the input file using the clip-regions
function, and then running LilyPond with the -dclip-systems option:

\layout {
  clip-regions
  = #(list
      (cons
        (make-rhythmic-location 5 1 2)
        (make-rhythmic-location 7 3 4)))
}

This example will extract a single fragment of the input file starting after a half-note duration
in fifth measure (5 1 2) and ending after the third quarter-note in the seventh measure (7 3 4).

Additional fragments can be extracted by adding more pairs of make-rhythmic-location
entries to the clip-regions list in the \layout block.

By default, each music fragment will be output as a separate EPS file, but other formats such
as PDF or PNG can also be created if required. The extracted music is output as if had been
literally ‘cut’ from the original printed score so if a fragment runs over one or more lines, a
separate output file for each line will be generated.

See also

Notation Reference: Section 4.2.1 [The layout block], page 557.

Application Usage: Section “Command-line usage” in Application Usage.
3.4.2 Skipping corrected music

When entering or copying music, usually only the music near the end (where new notes are being added) is interesting to view and correct. To speed up this correction process, it is possible to skip typesetting of all but the last few measures. This is achieved by defining a special variable at the source file’s top level, as follows:

```lilypond
showLastLength = R1*5
\score { ... }
```

In this instance, nothing will be rendered but the last five measures (assuming 4/4 time signature) of every `\score` in the input file. For longer pieces, rendering only a small part is often an order of magnitude quicker than rendering it completely. When working on the beginning of a score that has already been typeset (for example to add a new part), the `showFirstLength` property may be useful as well.

Skipping parts of a score can be controlled in a more fine-grained fashion with the property `Score.skipTypesetting`. When it is set, no typesetting is performed at all. As a property of the `Score` context, it affects all voices and staves; see [Score - the master of all contexts], page 598.

This property is also used to control output to the MIDI file. If some event in the skipped section alters some of its context properties, for example a tempo or instrument change, then that new setting will take effect only at the point in time where `skipTypesetting` is disabled again:

```lilypond
\relative c' { 
  c4 c c c
  \set Score.skipTypesetting = ##t
  d4 d d d
  \tempo 4 = 80
  e4 e e e
  \set Score.skipTypesetting = ##f
  f4 f f f
}
```

Predefined commands

`showLastLength`, `showFirstLength`.

See also

Notation Reference: Section 5.1 [Interpretation contexts], page 597, [Score - the master of all contexts], page 598, Section A.17 [All context properties], page 776.

3.4.3 Alternative output formats

The default output formats for the printed score are Portable Document Format (PDF) and PostScript (PS). Portable Network Graphics (PNG), Scalable Vector Graphics (SVG) and Encapsulated PostScript (EPS) output formats are available through command line options, see Section “Basic command line options for LilyPond” in Application Usage.
SVG Output
SVG output can optionally contain metadata for graphical objects (grobs) like note heads, rests, etc. This metadata can be standard SVG attributes like id and class, or non-standard custom attributes. Specify the attributes and their values by overriding a grob's output-attributes property with a Scheme association list (alist). The values can be numbers, strings, or symbols. For example:

```scheme
{ 
  \once \override NoteHead.output-attributes = 
  #'((id . 123) 
    (class . "this that") 
    (data-whatever . something))
}
```

The input above will produce the following `<g>` (group) tag in the SVG file:

```xml
<g id="123" class="this that" data-whatever="something">
  ...NoteHead grob SVG elements...
</g>
```

The `<g>` tag contains all of the SVG elements for a given grob. (Some grobs generate multiple SVG elements.) In SVG syntax the data- prefix is used for non-standard custom metadata attributes.

3.4.4 Replacing the notation font
Gonville is an alternative set of glyphs to Feta – part of the Emmentaler font – and used in LilyPond. They can be downloaded from:

http://www.chiark.greenend.org.uk/~sgtatham/gonville/ (http://www.chiark.greenend.org.uk/~sgtatham/gonville/)

Here are a few sample bars of music set in Gonville:
Here are a few sample bars of music set in LilyPond’s Feta glyphs:

Installation Instructions


Note: `gonville-*.otf` files are for ps and eps backend (for PDF and PostScript outputs). `gonville-*.svg` files are for svg backend without svg-woff option. `gonville-*.woff` files are for svg backend with svg-woff option. For more information, see Section “Advanced command line options for LilyPond” in Application Usage.

The following syntax changes the notation font (general and brace) to the Gonville font.

```lilypond
\paper {
  #(define fonts
    (set-global-fonts
      #:music "gonville"
      #:brace "gonville"
    ))
}
```

Note: Each call to set-global-fonts completely resets both the main notation and text fonts. If any category is left unspecified, then the default font will be used for that category. Each call to set-global-fonts changes the fonts for each \book that follows it, whether created explicitly or implicitly. This means that each \book can have its own set of main fonts by calling set-global-fonts before it. For more information, see [Entire document fonts], page 271.

See also


Notation Reference: Section A.8 [The Emmentaler font], page 683, [Entire document fonts], page 271.

Known issues and warnings

Gonville cannot be used to typeset ‘Ancient Music’ notation and it is likely newer glyphs in later releases of LilyPond may not exist in the Gonville font family. Please refer to the author’s website for more information on these and other specifics, including licensing of Gonville.
Other notation fonts

If you have other notation fonts like `fontname-*.otf`, `fontname-*.svg`, and `fontname-*.woff`, you can use them in the same way as Gonville.

That is, copy the `fontname-*.otf` files to `.../share/lilypond/current/fonts/otf` or `.../share/lilypond/X.Y.Z/fonts/otf`. If you have `fontname-*.svg` and `fontname-*.woff` files, copy them to `.../share/lilypond/current/fonts/svg` or `.../share/lilypond/X.Y.Z/fonts/svg`.

Note: At the moment, LilyPond expects the font file names to have the following suffixes, all of which must be present in the above installation folder(s) to work properly: -11, -13, -14, -16, -18, -20, -23, -26, -brace. For example, `emmentaler-11.otf`, `emmentaler-20.svg`, and `emmentaler-brace.woff` etc.

The following syntax changes the notation font (general and brace) to the `fontname` font.

```latex
\paper {
  #(define fonts
    (set-global-fonts
     #:music "fontname" ; font filename without suffix and extension
     #:brace "fontname" ; font filename without suffix and extension
    ))
}
```

Note: For `music` and `brace` categories, specify the font filename without the suffix and extension.

3.5 Creating MIDI output

LilyPond can produce files that conform to the MIDI (Musical Instrument Digital Interface) standard and so allow for the checking of the music output aurally (with the help of an application or device that understands MIDI). Listening to MIDI output may also help in spotting errors such as notes that have been entered incorrectly or are missing accidentals and so on.

MIDI files do not contain sound (like AAC, MP3 or Vorbis files) but require additional software to produce sound from them.

3.5.1 Supported notation for MIDI

The following musical notation can be used with LilyPond’s default capabilities to produce MIDI output:

- Breath marks
- Chords entered as chord names
- Crescendi, decrescendi over multiple notes. The volume is altered linearly between the two extremes
- Dynamic markings from `ppppp` to `fffff`, including `mp`, `mf` and `sf`
- Microtones but not microtonal chords. A MIDI player that supports pitch bending will also be required.
- Lyrics
- Pitches
- Rhythms entered as note durations, including tuplets
- ‘Simple’ articulations; staccato, staccatissimo, accent, marcato and portato
- Tempo changes using the \tempo function
- Ties
- Tremolos that are not entered with a ‘:`[number]’ value
Panning, balance, expression, reverb and chorus effects can also be controlled by setting context properties, see Section 3.5.8 [Context properties for MIDI effects], page 541.

When combined with the articulate script the following, additional musical notation can be output to MIDI:

- Appoggiaturas. These are made to take half the value of the note following (without taking dots into account). For example:
  \appoggiatura c8 d2.
  The c will take the value of a crotchet.
- Ornaments (i.e., mordents, trills and turns et al.)
- Rallentando, accelerando, ritardando and a tempo
- Slurs, including phrasing slurs
- Tenuto

See Section 3.5.9 [Enhancing MIDI output], page 542.

### 3.5.2 Unsupported notation for MIDI

The following items of musical notation cannot be output to MIDI:

- Articulations other than staccato, staccatissimo, accent, marcato and portato
- Crescendi and decrescendi over a single note
- Fermata
- Figured bass
- Glissandi
- Falls and doits
- Microtonal chords
- Rhythms entered as annotations, e.g., swing
- Tempo changes without \tempo (e.g., entered as annotations)
- Tremolos that are entered with a ‘:[number]’ value

### 3.5.3 The MIDI block

To create a MIDI output file from a LilyPond input file, insert a \midi block, which can be empty, within the \score block:

\score {
  ...
  \layout { }
  \midi { }
}

Note: A \score block that, as well as the music, contains only a \midi block (i.e., without the \layout block), will only produce MIDI output files. No notation will be printed.

The default output file extension (.midi) can be changed by using the -dmidi-extension option with the lilypond command:

```
lilypond -dmidi-extension=mid MyFile.ly
```

Alternatively, add the following Scheme expression before the start of either the \book, \bookpart or \score blocks. See Section 3.1.5 [File structure], page 490.

```
#(ly:set-option 'midi-extension "mid")
```
See also

Notation Reference: Section 3.1.5 [File structure], page 490, Section 3.2.3 [Creating output file metadata], page 504.

Installed Files: scm/midi.scm.

Known issues and warnings

There are fifteen MIDI channels available and one additional channel (#10) for drums. Staves are assigned to channels in sequence, so a score that contains more than fifteen staves will result in the extra staves sharing (but not overwriting) the same MIDI channel. This may be a problem if the sharing staves have conflicting, channel-based, MIDI properties – such as different MIDI instruments – set.

Using a \m inidi block with polymetric notation may cause unexpected barcheck warnings. In this case move the \Timing\_translator from the \Score context to the \Staff context within the \m inidi block.

\m inidi {
  \context {
    \Score
    \remove "Timing\_translator"
  }
  \context {
    \Staff
    \consists "Timing\_translator"
  }
}

3.5.4 Controlling MIDI dynamics

It is possible to control the overall MIDI volume, the relative volume of dynamic markings and the relative volume of different instruments.

Dynamic marks translate automatically into volume levels in the available MIDI volume range whereas crescendos and decrescendos vary the volume linearly between their two extremes. It is possible to control the relative volume of dynamic markings, and the overall volume levels of different instruments.

Dynamic marks in MIDI

Only the dynamic markings from $\text{pppp}$ to $\text{ffff}$, including $\text{mp}$, $\text{mf}$ and $\text{sf}$ have values assigned to them. This value is then applied to the value of the overall MIDI volume range to obtain the final volume included in the MIDI output for that particular dynamic marking. The default fractions range from 0.25 for $\text{pppp}$ to 0.95 for $\text{ffff}$. The complete set of dynamic marks and their associated fractions can be found in scm/midi.scm.

Selected Snippets

Creating custom dynamics in MIDI output

The following example shows how to create a dynamic marking, not included in the default list, and assign it a specific value so that it can be used to affect MIDI output.

The dynamic mark $\text{rfz}$ is assigned a value of 0.9.

\begin{verbatim}
#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
    0.9
    (default-dynamic-absolute-volume dynamic)))
\end{verbatim}
Setting MIDI volume

The minimum and maximum overall volume of MIDI dynamic markings is controlled by setting the properties `midiMinimumVolume` and `midiMaximumVolume` at the `Score` level. These properties have an effect only at the start of a voice and on dynamic marks. The fraction corresponding to each dynamic mark is modified with this formula:

\[
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) \times \text{fraction}
\]

In the following example the dynamic range of the overall MIDI volume is limited to the range 0.2 - 0.5.

\score {
\new Staff {
  \set Staff.midiInstrument = "flute"
  \relative {
    a'4\pp b c-\rfz
  }
}
\new Staff {
  \set Staff.midiInstrument = "clarinet"
  \relative {
    a'4\pp b c-\rfz
  }
}
\context {
  \Score
  midiMinimumVolume = #0.2
  midiMaximumVolume = #0.5
}
}
Simple MIDI instrument equalization can be achieved by setting `midiMinimumVolume` and `midiMaximumVolume` properties within the `Staff` context.

```lilypond
\score { 
  \new Staff { 
    \set Staff.midiInstrument = "flute"
    \set Staff.midiMinimumVolume = #0.7
    \set Staff.midiMaximumVolume = #0.9
    ... music ...
  } 
  \midi { } 
}
```

For scores with multiple staves and multiple MIDI instruments, the relative volumes of each instrument can be set individually:

```lilypond
\score { 
  \new Staff { 
    \set Staff.midiInstrument = "flute"
    \set Staff.midiMinimumVolume = #0.7
    \set Staff.midiMaximumVolume = #0.9
    ... music ...
  } 
  \new Staff { 
    \set Staff.midiInstrument = "clarinet"
    \set Staff.midiMinimumVolume = #0.3
    \set Staff.midiMaximumVolume = #0.6
    ... music ...
  } 
  \midi { } 
}
```

In this example the volume of the clarinet is reduced relative to the volume of the flute.

If these volumes properties are not set then LilyPond still applies a ‘small degree’ of equalization to certain instruments. See `scm/midi.scm`.

Installed Files: `scm/midi.scm`.

See also

Notation Reference: Section 4.2 [Score layout], page 556.


Selected Snippets

Replacing default MIDI instrument equalization

The default MIDI instrument equalizer can be replaced by setting the `instrumentEqualizer` property in the `Score` context to a user-defined Scheme procedure that uses a MIDI instrument name as its argument along with a pair of fractions indicating the minimum and maximum volumes respectively to be applied to that specific instrument.

The following example sets the minimum and maximum volumes for flute and clarinet respectively.

```lilypond
#(define my-instrument-equalizer-alist '())
```
\begin{verbatim}
(set! my-instrument-equalizer-alist
  (append
   '(
     ("flute" . (0.7 . 0.9))
     ("clarinet" . (0.3 . 0.6)))
     my-instrument-equalizer-alist))

(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  \new Staff {
    \key g \major
    \time 2/2
    \set Score.instrumentEqualizer = #my-instrument-equalizer
    \set Staff.midiInstrument = "flute"
    \new Voice \relative {
      r2 g'' \mp g fis~
      4 g8 fis e2~
      4 d8 cis d2
    }
  }
  \new Staff {
    \key g \major
    \set Staff.midiInstrument = "clarinet"
    \new Voice \relative {
      b'1 \p a2. b8 a
      g2. fis8 e
      fis2 r
    }
  }
}

\layout { }
\midi { }
\end{verbatim}

Known issues and warnings
Changes in the MIDI volume take place only on starting a note, so crescendi and decrescendi cannot affect the volume of a single note.
Setting MIDI block properties

The `\midi` block can contain context rearrangements, new context definitions or code that sets the values of certain properties.

\[\begin{align*}
\text{\score} & \{ \\
& \hspace{1em} \text{... music ...} \\
& \hspace{1em} \text{\midi} \{ \\
& \hspace{2em} \text{\tempo 4 = 72} \\
& \} \\
& \} \\
\end{align*}\]

Here the tempo is set to 72 quarter-note beats per minute. The tempo mark in the `\midi` block will not appear in the printed score. Although any other `\tempo` indications specified within the `\score` block will also be reflected in the MIDI output.

In a `\midi` block the `\tempo` command is setting properties during the interpretation of the music and in the context of output definitions; so it is interpreted as if it were a context modification.

Context definitions follow the same syntax as those in a `\layout` block;

\[\begin{align*}
\text{\score} & \{ \\
& \hspace{1em} \text{... music ...} \\
& \hspace{1em} \text{\midi} \{ \\
& \hspace{2em} \text{\context} \{ \\
& \hspace{3em} \text{\Voice} \\
& \hspace{4em} \text{\remove "Dynamic_performer"} \\
& \} \\
& \} \\
& \} \\
\end{align*}\]

This example removes the effect of dynamics from the MIDI output. Note: LilyPond’s translation modules used for sound are called ‘performers’.

See also


Notation Reference: Section 1.3 [Expressive marks], page 122, Section 4.2 [Score layout], page 556.

Installed Files: ly/performer-init.ly.

Snippets: Section “MIDI” in Snippets.


Known issues and warnings

Some MIDI players do not always correctly handle tempo changes in the MIDI output.

Changes to the `\midiInstrument`, as well as some MIDI options, at the beginning of a staff may appear twice in the MIDI output.

3.5.5 Using MIDI instruments

MIDI instruments are set using the `\midiInstrument` property within a `Staff` context.

\[\begin{align*}
\text{\score} & \{ \\
& \hspace{1em} \text{\new Staff} \{ \\
& \hspace{2em} \text{\set Staff.midiInstrument = "glockenspiel"} \\
& \hspace{2em} \text{... music ...} \\
& \} \\
\end{align*}\]
\midi { }
}

or
\score {
  \new Staff \with {midiInstrument = "cello"} {
    ... music ...
  }
  \midi { }
}

If the instrument name does not match any of the instruments listed in the ‘MIDI instruments’ section, the acoustic grand instrument will be used instead. See Section A.6 [MIDI instruments], page 680.

See also


Notation Reference: Section A.6 [MIDI instruments], page 680, Section 4.2 [Score layout], page 556.

Installed Files: scm/midi.scm.

Known issues and warnings

Percussion instruments that are notated in a DrumStaff context will be output, correctly, to MIDI channel 10 but some pitched, percussion instruments like the xylophone, marimba, vibraphone or timpani, are treated as “normal” instruments so the music for these should be entered in a Staff (not DrumStaff) context to obtain correct MIDI output. A full list of channel 10 drum-kits entries can be found in scm/midi.scm. See Section “Other sources of information” in Learning Manual.

3.5.6 Using repeats with MIDI

Repeats can be represented in the MIDI output by applying the \unfoldRepeats command.

\score {
  \unfoldRepeats {
    \repeat tremolo 8 { c'32 e' }
    \repeat percent 2 { c''8 d'' }
    \repeat volta 2 { c'4 d' e' f' }
    \alternative {
      { g' a' a' g' }
      { f' e' d' c' }
    }
  }
  \midi { }
}

In order to restrict the effect of \unfoldRepeats to the MIDI output only, while also generating printable scores, it is necessary to make two \score blocks; one for MIDI (with unfolded repeats) and one for the notation (with volta, tremolo, and percent repeats);

\score {
  ... music ...
  \layout { }
}
\score {
  \unfoldRepeats {

When using multiple voices, each of the voices must contain completely unfolded repeats for correct MIDI output.

See also

Notation Reference: Section 1.4 [Repeats], page 151.

### 3.5.7 MIDI channel mapping

When generating a MIDI file from a score, LilyPond will automatically assign every note in the score to a MIDI channel, the one on which it should be played when it is sent to a MIDI device. A MIDI channel has a number of controls available to select, for example, the instrument to be used to play the notes on that channel, or to request the MIDI device to apply various effects to the sound produced on the channel. At all times, every control on a MIDI channel can have only a single value assigned to it (which can be modified, however, for example, to switch to another instrument in the middle of a score).

The MIDI standard supports only 16 channels per MIDI device. This limit on the number of channels also limits the number of different instruments which can be played at the same time.

LilyPond creates separate MIDI tracks for each staff, (or discrete instrument or voice, depending on the value of `Score.midiChannelMapping`), and also for each lyrics context. There is no limit to the number of tracks.

To work around the limited number of MIDI channels, LilyPond supports a number of different modes for MIDI channel allocation, selected using the `Score.midiChannelMapping` context property. In each case, if more MIDI channels than the limit are required, the allocated channel numbers wrap around back to 0, possibly causing the incorrect assignment of instruments to some notes. This context property can be set to one of the following values:

**'staff**

Allocate a separate MIDI channel to each staff in the score (this is the default). All notes in all voices contained within each staff will share the MIDI channel of their enclosing staff, and all are encoded in the same MIDI track.

The limit of 16 channels is applied to the total number of staff and lyrics contexts, even though MIDI lyrics do not take up a MIDI channel.

**'instrument**

Allocate a separate MIDI channel to each distinct MIDI instrument specified in the score. This means that all the notes played with the same MIDI instrument will share the same MIDI channel (and track), even if the notes come from different voices or staves.

In this case the lyrics contexts do not count towards the MIDI channel limit of 16 (as they will not be assigned to a MIDI instrument), so this setting may allow a better allocation of MIDI channels when the number of staves and lyrics contexts in a score exceeds 16.

**'voice**

Allocate a separate MIDI channel to each voice in the score that has a unique name among the voices in its enclosing staff. Voices in different staves are always assigned separate MIDI channels, but any two voices contained within the same staff will share the same MIDI channel if they have the same name. Because `midiInstrument`
and the several MIDI controls for effects are properties of the staff context, they
cannot be set separately for each voice. The first voice will be played with the
instrument and effects specified for the staff, and voices with a different name from
the first will be assigned the default instrument and effects.

Note: different instruments and/or effects can be assigned to several voices on the
same staff by moving the \Staff_performer from the \Staff to the \Voice context,
and leaving midiChannelMapping to default to 'staff' or set to 'instrument'; see
the snippet below.

For example, the default MIDI channel mapping of a score can be changed to the 'instrument
setting as shown:

\score { 
\...music...
\midi {
\context { 
\Score 
midiChannelMapping = #'instrument 
}\}
}\}

**Selected Snippets**

*Changing MIDI output to one channel per voice*

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel,
with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI
channels, since there are only 16 available per track.

However, by moving the \Staff_performer to the \Voice context, each voice on a staff can
have its own MIDI channel, as is demonstrated by the following example: despite being on the
same staff, two MIDI channels are created, each with a different \midiInstrument.

\score { 
\new Staff << 
\new Voice \relative c''' { 
\set midiInstrument = #'flute
\voiceOne 
\key g \major 
\time 2/2 
r2 g-"Flute" ~
g fis ~
fis4 g8 fis e2 ~
e4 d8 cis d2 
}
\new Voice \relative c'' { 
\set midiInstrument = #'clarinet
\voiceTwo 
b1-"Clarinet"
a2. b8 a 
g2. fis8 e 
fis2 r
}
>> 
\layout { }

### 3.5.8 Context properties for MIDI effects

The following context properties can be used to apply various MIDI effects to notes played on the MIDI channel associated with the current staff, MIDI instrument or voice (depending on the value of the `Score.midiChannelMapping` context property and the context in which the `Staff_performer` is located; see Section 3.5.7 [MIDI channel mapping], page 539).

Changing these context properties will affect all notes played on the channel after the change, however some of the effects may even apply also to notes which are already playing (depending on the implementation of the MIDI output device).

The following context properties are supported:

**Staff.midiPanPosition**

The pan position controls how the sound on a MIDI channel is distributed between left and right stereo outputs. The context property accepts a number between -1.0 (`#LEFT`) and 1.0 (`#RIGHT`); the value -1.0 will put all sound power to the left stereo output (keeping the right output silent), the value 0.0 (`#CENTER`) will distribute the sound evenly between the left and right stereo outputs, and the value 1.0 will move all sound to the right stereo output. Values between -1.0 and 1.0 can be used to obtain mixed distributions between left and right stereo outputs.

**Staff.midiBalance**

The stereo balance of a MIDI channel. Similarly to the pan position, this context property accepts a number between -1.0 (`#LEFT`) and 1.0 (`#RIGHT`). It varies the relative volume sent to the two stereo speakers without affecting the distribution of the stereo signals.

**Staff.midiExpression**

Expression level (as a fraction of the maximum available level) to apply to a MIDI channel. A MIDI device combines the MIDI channel’s expression level with a voice’s current dynamic level (controlled using constructs such as \\p or \\ff) to obtain the total volume of each note within the voice. The expression control could be used, for example, to implement crescendo or decrescendo effects over single sustained notes (not supported automatically by LilyPond).

The expression level ranges from 0.0 (no expression, meaning zero volume) to 1.0 (full expression).
Staff.midiReverbLevel

Reverb level (as a fraction of the maximum available level) to apply to a MIDI channel. This property accepts numbers between 0.0 (no reverb) and 1.0 (full effect).

Staff.midiChorusLevel

Chorus level (as a fraction of the maximum available level) to apply to a MIDI channel. This property accepts numbers between 0.0 (no chorus effect) and 1.0 (full effect).

Known issues and warnings

As MIDI files do not contain any actual audio data, changes in these context properties translate only to requests for changing MIDI channel controls in the outputted MIDI files. Whether a particular MIDI device (such as a software MIDI player) can actually handle any of these requests in a MIDI file is entirely up to the implementation of the device: a device may choose to ignore some or all of these requests. Also, how a MIDI device will interpret different values for these controls (generally, the MIDI standard fixes the behavior only at the endpoints of the value range available for each control), and whether a change in the value of a control will affect notes already playing on that MIDI channel or not, is also specific to the MIDI device implementation.

When generating MIDI files, LilyPond will simply transform the fractional values within each range linearly into values in a corresponding (7-bit, or 14-bit for MIDI channel controls which support fine resolution) integer range (0-127 or 0-32767, respectively), rounding fractional values towards the nearest integer away from zero. The converted integer values are stored as-is in the generated MIDI file. Please consult the documentation of your MIDI device for information about how the device interprets these values.

3.5.9 Enhancing MIDI output

The default MIDI output is basic but can be improved by setting MIDI instruments and various \midi block properties.

Additional scripts allow to fine-tune the way dynamics, articulations and rhythm are rendered in MIDI: the articulate script and the swing script.

The articulate script

To use the articulate script add the appropriate \include command at the top of the input file:

\include "articulate.ly"

The script creates MIDI output into appropriately ‘time-scaled’ notes to match many articulation and tempo indications. Engraved output however, will also be altered to literally match the MIDI output.

\score {
   \articulate {
      ... music ...
   }
   \midi { }
}

The \articulate command enables abbreviations (such as trills and turns) to be processed. A full list of supported items can be found in the script itself. See ly/articulate.ly.

See also


Notation Reference: Section 4.2 [Score layout], page 556.
Chapter 3: General input and output

Installed Files: ly/articulate.ly.

Note: The articulate script may shorten chords, which might not be appropriate for some types of instrument, such as organ music. Notes that do not have any articulations attached to them may also be shortened; so to allow for this, restrict the use of the \articulate function to shorter segments of music, or modify the values of the variables defined in the articulate script to compensate for the note-shortening behavior.

The swing script

The swing script provides additional functions allowing for regular durations to be played with an unequal rhythm. The most obvious example is 'swing' interpretation commonly found in jazz music where binary eighth notes should be played in a ternary fashion; however additional interpretations are also supported.

This script has to be \include-d at the beginning of the input file:

\include "swing.ly"

Three commands are provided:

- \tripletFeel creates a triplet-feel swing. It takes two arguments: the durations that should be affected by it (typically 8 for eighth notes), and then the music expression to which it should be applied.

- \applySwing takes an additional argument prior to the music expression: a 'weight list' of n number ratios expressing the way regular notes should be played: for example, #'(2 1) indicates that every other note should be played twice as long as the following note (in fact, \tripletFeel duration {music} is actually a shortcut for \applySwing duration #'(2 1) {music}). Smoother swung eighths may be obtained with a weight list of #'(3 2), or other values depending on taste.

That list may include more than two values, which allows for longer and more sophisticated groove patterns; for example, a samba feel for sixteenth notes may be obtained as follows:

\score {
  \applySwing 16 #'(3 2 2 3) {
    ... music ...
  }
  \midi { }
}

- \applySwingWithOffset adds yet another argument between the 'weight list' and the music expression: an offset length (entered as a ly:make-moment expression). This command should be used when the music expression has to start off-beat, with a partial swing cycle.

Note: As with the articulate script, all swing commands are also rendered in the engraved output, which results in irregular note spacing. This can be avoided by using them only in a \score block dedicated to MIDI output, rather than to printed music.

Additional help and information is included in the script file: see ly/swing.ly.

See also

Notation Reference: Section 1.2 [Rhythms], page 46.
**Installed Files:** `ly/swing.ly`.

**Known issues and warnings**

- `\repeat` constructs in music (even `\repeat unfold`) are not taken into consideration when determining note timing. This will lead to problems unless the durations of all repeated parts are integer multiples of the swing cycle duration.
- These functions are oblivious to time signatures and measures. That is why offsets need to be supplied by using `\applySwingWithOffset` if music starts off-beat.
- Grace notes are ignored and simply left unaffected; so are tuplets.

### 3.6 Extracting musical information

In addition to creating graphical output and MIDI, LilyPond can display musical information as text.

#### 3.6.1 Displaying LilyPond notation

Displaying a music expression in LilyPond notation can be done with the music function `\displayLilyMusic`. To see the output, you will typically want to call LilyPond using the command line. For example,

```
{ \displayLilyMusic \transpose c a, \{ c4 e g a bes \} }
```

will display

```
\{ a,4 cis4 e4 fis4 g4 \}
```

By default, LilyPond will print these messages to the console along with all the other LilyPond compilation messages. To split up these messages and save the results of `\displayLilyMusic`, redirect the output to a file.

```
lilypond file.ly >display.txt
```

Note that LilyPond does not just display the music expression, but also interprets it (since `\displayLilyMusic` returns it in addition to displaying it). Just insert `\displayLilyMusic` into the existing music in order to get information about it.

To interpret and display a music section in the console but, at the same time, remove it from the output file use the `\void` command.

```
{ \void \displayLilyMusic \transpose c a, \{ c4 e g a bes \} }
```

#### 3.6.2 Displaying scheme music expressions

See Section “Displaying music expressions” in *Extending*.

#### 3.6.3 Saving music events to a file

Music events can be saved to a file on a per-staff basis by including a file in your main score.

```
\include "event-listener.ly"
```

This will create file(s) called `FILENAME-STAFFNAME.notes` or `FILENAME-unnamed-staff.notes` for each staff. Note that if you have multiple unnamed staves, the events for all staves will be mixed together in the same file. The output looks like this:

```
0.000 note 57 4 p-c 2 12
```
0.000  dynamic  f
0.250  note    62    4  p-c 7 12
0.500  note    66    8  p-c 9 12
0.625  note    69    8  p-c 14 12
0.750  rest    4
0.750  breathe

The syntax is a tab-delimited line, with two fixed fields on each line followed by optional parameters.

    time  type  ...params...

This information can easily be read into other programs such as python scripts, and can be very useful for researchers wishing to perform musical analysis or playback experiments with LilyPond.

**Known issues and warnings**

Not all lilypond music events are supported by `event-listener.ly`. It is intended to be a well-crafted “proof of concept”. If some events that you want to see are not included, copy `event-listener.ly` into your lilypond directory and modify the file so that it outputs the information you want.
4 Spacing issues

The global paper layout is determined by three factors: the page layout, the line breaks, and the spacing. These all influence each other. The choice of spacing determines how densely each system of music is set. This influences where line breaks are chosen, and thus ultimately, how many pages a piece of music takes.

Globally speaking, this procedure happens in four steps: first, flexible distances (‘springs’) are chosen, based on durations. All possible line breaking combinations are tried, and a ‘badness’ score is calculated for each. Then the height of each possible system is estimated. Finally, a page breaking and line breaking combination is chosen so that neither the horizontal nor the vertical spacing is too cramped or stretched.

Two types of blocks can contain layout settings: \texttt{\textbackslash paper \{\ldots\}} and \texttt{\textbackslash layout \{\ldots\}}. The \texttt{\textbackslash paper} block contains page layout settings that are expected to be the same for all scores in a book or bookpart, such as the paper height, or whether to print page numbers, etc. See Section 4.1 [Page layout], page 546. The \texttt{\textbackslash layout} block contains score layout settings, such as the number of systems to use, or the space between staff-groups, etc. See Section 4.2 [Score layout], page 556.

4.1 Page layout

This section discusses page layout options for the \texttt{\textbackslash paper} block.

4.1.1 The \texttt{\textbackslash paper} block

\texttt{\textbackslash paper} blocks may be placed in three different places to form a descending hierarchy of \texttt{\textbackslash paper} blocks:

- At the top of the input file, before all \texttt{\textbackslash book}, \texttt{\textbackslash bookpart}, and \texttt{\textbackslash score} blocks.
- Within a \texttt{\textbackslash book} block but outside all the \texttt{\textbackslash bookpart} and \texttt{\textbackslash score} blocks within that book.
- Within a \texttt{\textbackslash bookpart} block but outside all \texttt{\textbackslash score} blocks within that bookpart.

A \texttt{\textbackslash paper} block cannot be placed within a \texttt{\textbackslash score} block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy.

Several \texttt{\textbackslash paper} blocks can appear at each of the levels, for example as parts of several \texttt{\textbackslash included} files. If so, the fields at each level are merged, with values encountered last taking precedence if duplicated fields appear.

Settings that can appear in a \texttt{\textbackslash paper} block include:

- the \texttt{set-paper-size} scheme function,
- \texttt{\textbackslash paper} variables used for customizing page layout, and
- markup definitions used for customizing the layout of headers, footers, and titles.

The \texttt{set-paper-size} function is discussed in the next section, Section 4.1.2 [Paper size and automatic scaling], page 547. The \texttt{\textbackslash paper} variables that deal with page layout are discussed in later sections. The markup definitions that deal with headers, footers, and titles are discussed in Section 3.2.2 [Custom titles headers and footers], page 499.

Most \texttt{\textbackslash paper} variables will only work in a \texttt{\textbackslash paper} block. The few that will also work in a \texttt{\textbackslash layout} block are listed in Section 4.2.1 [The \texttt{\textbackslash layout} block], page 557.

Except when specified otherwise, all \texttt{\textbackslash paper} variables that correspond to distances on the page are measured in millimeters, unless a different unit is specified by the user. For example, the following declaration sets \texttt{top-margin} to ten millimeters:

\begin{verbatim}
\textbackslash paper \{
\end{verbatim}
top-margin = 10
}

To set it to 0.5 inches, use the \in unit suffix:

\paper {
  top-margin = 0.5\in
}

The available unit suffixes are \mm, \cm, \in, and \pt. These units are simple values for converting from millimeters; they are defined in ly/paper-defaults-init.ly. For the sake of clarity, when using millimeters, the \mm is typically included in the code, even though it is not technically necessary.

It is also possible to define \paper values using Scheme. The Scheme equivalent of the above example is:

\paper {
  #(define top-margin (* 0.5 in))
}

See also

Notation Reference: Section 4.1.2 [Paper size and automatic scaling], page 547, Section 3.2.2 [Custom titles headers and footers], page 499, Section 4.2.1 [The \layout block], page 557.


4.1.2 Paper size and automatic scaling

Setting the paper size

‘A4’ is the default value when no explicit paper size is set. However, there are two functions that can be used to change it:

set-default-paper-size
  #(set-default-paper-size "quarto")
  which must always be placed at the toplevel scope, and

set-paper-size
  \paper {
    #(set-paper-size "tabloid")
  }
  which must always be placed in a \paper block.

If the set-default-paper-size function is used in the toplevel scope, it must come before any \paper block. set-default-paper-size sets the paper size for all pages, whereas set-paper-size only sets the paper size for the pages that the \paper block applies to. For example, if the \paper block is at the top of the file, then it will apply the paper size to all pages. If the \paper block is inside a \book, then the paper size will only apply to that book.

When the set-paper-size function is used, it must be placed before any other functions used within the same \paper block. See [Automatic scaling to paper size], page 548.

Paper sizes are defined in scm/paper.scm, and while it is possible to add custom sizes, they will be overwritten on subsequent software updates. The available paper sizes are listed in Section A.5 [Predefined paper sizes], page 677.

The following command can be used in the file to add a custom paper size which can then be used with set-default-paper-size or set-paper-size as appropriate,

#(set! paper-alist (cons '("my size" . (cons (* 15 in) (* 3 in))) paper-alist))

paper {  
  #(set-paper-size "my size") 
}

The units in (inches), cm (centimeters) and mm (millimeters) can all be used.

If the symbol 'landscape' is added to the paper size function, pages will be rotated by 90
degrees, and wider line widths will be set accordingly.

#(set-default-paper-size "a6" 'landscape)

Swapping the paper dimensions without having the print rotated (like when printing to
postcard size, or creating graphics for inclusion rather than a standalone document) can be
achieved by appending 'landscape' to the name of the paper size itself:

#(set-default-paper-size "a6landscape")

When the paper size ends with an explicit 'landscape' or 'portrait', the presence of a
'landscape' symbol only affects print orientation, not the paper dimensions used for layout.

See also

Notation Reference: [Automatic scaling to paper size], page 548, Section A.5 [Predefined
paper sizes], page 677.

 Installed Files: scm/paper.scm.

Automatic scaling to paper size

If the paper size is changed with one of the scheme functions (set-default-paper-size or
set-paper-size), the values of several \paper variables are automatically scaled to the new
size. To bypass the automatic scaling for a particular variable, set the variable after setting the
paper size. Note that the automatic scaling is not triggered by setting the paper-height or
paper-width variables, even though paper-width can influence other values (this is separate
from scaling and is discussed below). The set-default-paper-size and set-paper-size
functions are described in [Setting the paper size], page 547.

The vertical dimensions affected by automatic scaling are top-margin and bottom-margin
(see Section 4.1.3 [Fixed vertical spacing \paper variables], page 548). The horizontal dimensions
affected by automatic scaling are left-margin, right-margin, inner-margin, outer-margin,
binding-offset, indent, and short-indent (see Section 4.1.5 [Horizontal spacing \paper
variables], page 551).

The default values for these dimensions are set in ly/paper-defaults-init.ly, using internal
variables named top-margin-default, bottom-margin-default, etc. These are the values
that result at the default paper size a4. For reference, with a4 paper the paper-height is
297\mm and the paper-width is 210\mm.

See also

Notation Reference: Section 4.1.3 [Fixed vertical spacing \paper variables], page 548,
Section 4.1.5 [Horizontal spacing \paper variables], page 551.


4.1.3 Fixed vertical spacing \paper variables

Note: Some \paper dimensions are automatically scaled to the paper
size, which may lead to unexpected behavior. See [Automatic scaling
to paper size], page 548.
Chapter 4: Spacing issues

Default values (before scaling) are defined in \ly/paper-defaults-init.ly.

**paper-height**
The height of the page, unset by default. Note that the automatic scaling of some vertical dimensions is not affected by this.

**top-margin**
The margin between the top of the page and the top of the printable area. If the paper size is modified, this dimension’s default value is scaled accordingly.

**bottom-margin**
The margin between the bottom of the printable area and the bottom of the page. If the paper size is modified, this dimension’s default value is scaled accordingly.

**ragged-bottom**
If this is set to true, systems will be set at their natural spacing, neither compressed nor stretched vertically to fit the page.

**ragged-last-bottom**
If this is set to false, then the last page, and the last page in each section created with a \bookpart block, will be vertically justified in the same way as the earlier pages.

See also

Notation Reference: [Automatic scaling to paper size], page 548.

Installed Files: \ly/paper-defaults-init.ly.

Snippets: Section “Spacing” in Snippets.

Known issues and warnings

The titles (from the \header block) are treated as a system, so **ragged-bottom** and **ragged-last-bottom** will add space between the titles and the first system of the score.

Explicitly defined paper-sizes will override any user-defined top or bottom margin settings.

4.1.4 Flexible vertical spacing \paper variables

In most cases, it is preferable for the vertical distances between certain items (such as margins, titles, systems, and separate scores) to be flexible, so that they stretch and compress nicely according to each situation. A number of \paper variables (listed below) are available to fine-tune the stretching behavior of these dimensions.

Note that the \paper variables discussed in this section do not control the spacing of staves within individual systems. Within-system spacing is controlled by grob properties, with settings typically entered inside a \score or \layout block, and not inside a \paper block. See Section 4.4.1 [Flexible vertical spacing within systems], page 567.

Structure of flexible vertical spacing alists

Each of the flexible vertical spacing \paper variables is an alist (association list) containing four 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. The reference point of a (title or top-level) markup is its highest point, and the reference point of a system is the vertical center of the nearest StaffSymbol – even if a non-staff line (such as a Lyrics context) is in the way. Values for **basic-distance** that are less than either **padding** or **minimum-distance** are not meaningful, since the resulting distance will never be less than either **padding** or **minimum-distance**.
Chapter 4: Spacing issues

- **minimum-distance** – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect. Values for minimum-distance that are less than padding are not meaningful, since the resulting distance will never be less than padding.

- **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). When positive, the significance of a particular dimension’s stretchability value lies only in its relation to the stretchability values of the other dimensions. For example, if one dimension has twice the stretchability of another, it will stretch twice as easily. Values should be non-negative and finite. The value +inf.0 triggers a programming_error and is ignored, but 1.0e7 can be used for an almost infinitely stretchable spring. If unset, the default value is set to basic-distance. Note that the dimension’s propensity to compress cannot be directly set by the user and is equal to (basic-distance − minimum-distance).

If a page has a ragged bottom, the resulting distance is the largest of:

- basic-distance,
- minimum-distance, and
- padding plus the smallest distance necessary to eliminate collisions.

For multi-page scores with a ragged bottom on the last page, the last page uses the same spacing as the preceding page, provided there is enough space for that.

Specific methods for modifying alists are discussed in Section 5.3.7 [Modifying alists], page 631. The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely redefines the variable:

```
paper {
    system-system-spacing.basic-distance = #8
    score-system-spacing = #'(basic-distance . 12)
    (minimum-distance . 6)
    (padding . 1)
    (stretchability . 12))
}
```

**List of flexible vertical spacing \paper variables**

The names of these variables follow the format upper-lower-spacing, where upper and lower are the items to be spaced. Each distance is measured between the reference points of the two items (see the description of the alist structure above). Note that in these variable names, the term ‘markup’ refers to both title markups (bookTitleMarkup or scoreTitleMarkup) and top-level markups (see Section 3.1.5 [File structure], page 490). All distances are measured in staff-spaces.

Default settings are defined in ly/paper-defaults-init.ly.

- **markup-system-spacing**
  the distance between a (title or top-level) markup and the system that follows it.

- **score-markup-spacing**
  the distance between the last system of a score and the (title or top-level) markup that follows it.
Chapter 4: Spacing issues

4.1.5 Horizontal spacing \texttt{\textbackslash paper} variables

| Note: Some \texttt{\textbackslash paper} dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], page 548. |

\texttt{\textbackslash paper} variables for widths and margins

Default values (before scaling) that are not listed here are defined in \texttt{ly/paper-defaults-init.ly}.

\texttt{\textbackslash paper-width}

The width of the page, unset by default. While \texttt{\textbackslash paper-width} has no effect on the automatic scaling of some horizontal dimensions, it does influence the \texttt{\textbackslash line-width} variable. If both \texttt{\textbackslash paper-width} and \texttt{\textbackslash line-width} are set, then \texttt{\textbackslash left-margin} and \texttt{\textbackslash right-margin} will also be updated. Also see check-consistency.

\texttt{\textbackslash line-width}

When specified in a \texttt{\textbackslash paper} block this defines the horizontal extent available for the staff lines in un-indented systems. If left unspecified, the paper’s \texttt{\textbackslash line-width} is determined from \texttt{(\textbackslash paper-width \textbackslash subtraction \textbackslash left-margin \textbackslash subtraction \textbackslash right-margin)}. If the paper’s \texttt{\textbackslash line-width} is specified, and both \texttt{\textbackslash left-margin} and \texttt{\textbackslash right-margin} are not, then the margins will be updated to center the systems on the page automatically. Also see check-consistency.

\texttt{\textbackslash line-widths} for individual scores can be specified in the scores’ \texttt{\textbackslash layout} blocks. These values control the width of the lines produced on a score-by-score basis.
If line-width is not specified for a score, it defaults to the paper’s line-width. Setting a score’s line-width has no effect on the paper margins. Staff lines, of a length determined by the score’s line-width, are left-aligned within the paper area defined by the paper’s line-width. If the score and paper line-widths are equal, the staff lines will extend exactly from the left margin to the right margin, but if the score’s line-width is greater than the paper’s line-width the staff lines will run over into the right margin.

left-margin
The margin between the left edge of the page and the start of the staff lines in unindented systems. If the paper size is modified, this dimension’s default value is scaled accordingly. If left-margin is unset, and both line-width and right-margin are set, then left-margin is set to (paper-width – line-width – right-margin). If only line-width is set, then both margins are set to ((paper-width – line-width) / 2), and the systems are consequently centered on the page. Also see check-consistency.

right-margin
The margin between the right edge of the page and the end of the staff lines in non-ragged systems. If the paper size is modified, this dimension’s default value is scaled accordingly. If right-margin is unset, and both line-width and left-margin are set, then right-margin is set to (paper-width – line-width – left-margin). If only line-width is set, then both margins are set to ((paper-width – line-width) / 2), and the systems are consequently centered on the page. Also see check-consistency.

check-consistency
If this is true (the default value), print a warning if left-margin, line-width, and right-margin do not exactly add up to paper-width, and replace each of these (except paper-width) with their default values (scaled to the paper size if necessary). If set to false, ignore any inconsistencies and allow systems to run off the edge of the page.

ragged-right
If set to true, systems will not fill the line width. Instead, systems end at their natural horizontal length. Default: #t for scores with only one system, and #f for scores with two or more systems. This variable can also be set in a \layout block.

ragged-last
If set to true, the last system in the score will not fill the line width. Instead the last system ends at its natural horizontal length. Default: #f. This variable can also be set in a \layout block.

See also
Notation Reference: [Automatic scaling to paper size], page 548.


Known issues and warnings
Explicitly defined paper-sizes will override any user-defined left or right margin settings.

\paper variables for two-sided mode
Default values (before scaling) are defined in ly/paper-defaults-init.ly.
two-sided
If set to true, use inner-margin, outer-margin and binding-offset to determine margins depending on whether the page number is odd or even. This overrides left-margin and right-margin.

inner-margin
The margin all pages have at the inner side if they are part of a book. If the paper size is modified, this dimension’s default value is scaled accordingly. Works only with two-sided set to true.

outer-margin
The margin all pages have at the outer side if they are part of a book. If the paper size is modified, this dimension’s default value is scaled accordingly. Works only with two-sided set to true.

binding-offset
The amount inner-margin is increased to make sure nothing will be hidden by the binding. If the paper size is modified, this dimension’s default value is scaled accordingly. Works only with two-sided set to true.

See also
Notation Reference: [Automatic scaling to paper size], page 548.

\paper variables for shifts and indents
Default values (before scaling) that are not listed here are defined in ly/paper-defaults-init.ly.

horizontal-shift
The amount that all systems (including titles and system separators) are shifted to the right. Default: 0.0\mm.

indent
The level of indentation for the first system in a score. If the paper size is modified, this dimension’s default value is scaled accordingly. The space within line-width available for the first system is reduced by this amount. indent may also be specified in \layout blocks to set indents on a score-by-score basis.

short-indent
The level of indentation for all systems in a score besides the first system. If the paper size is modified, this dimension’s default value is scaled accordingly. The space within line-width available for systems other than the first is reduced by this amount. short-indent may also be specified in \layout blocks to set short indents on a score-by-score basis.

See also
Notation Reference: [Automatic scaling to paper size], page 548.
Snippets: Section “Spacing” in Snippets.

4.1.6 Other \paper variables
\paper variables for line breaking

**max-systems-per-page**
The maximum number of systems that will be placed on a page. This is currently supported only by the ly:optimal-breaking algorithm. Default: unset.

**min-systems-per-page**
The minimum number of systems that will be placed on a page. This may cause pages to be overfilled if it is made too large. This is currently supported only by the ly:optimal-breaking algorithm. Default: unset.

**systems-per-page**
The number of systems that should be placed on each page. This is currently supported only by the ly:optimal-breaking algorithm. Default: unset.

**system-count**
The number of systems to be used for a score. Default: unset. This variable can also be set in a \layout block.

See also
Notation Reference: Section 4.3.1 [Line breaking], page 560.

\paper variables for page breaking

Default values not listed here are defined in ly/paper-defaults-init.ly

**page-breaking**

**page-breaking-system-system-spacing**
Tricks the page breaker into thinking that system-system-spacing is set to something different than it really is. For example, if page-breaking-system-system-spacing.padding is set to something substantially larger than system-system-spacing.padding, then the page-breaker will put fewer systems on each page. Default: unset.

**page-count**
The number of pages to be used for a score, unset by default.

The following variables are effective only when page-breaking is set to ly:page-turn-breaking. Page breaks are then chosen to minimize the number of page turns. Since page turns are required on moving from an odd-numbered page to an even-numbered one, a layout in which the last page is odd-numbered will usually be favoured. Places where page turns are preferred can be indicated manually by inserting allowPageTurn or automatically by including the Page_turn_engraver (see [Optimal page turning], page 565).

If there are insufficient choices available for making suitable page turns, LilyPond may insert a blank page either within a score, between scores (if there are two or more scores), or by ending a score on an even-numbered page. The values of the following three variables may be increased to make these actions less likely.

The values are penalties, i.e., the higher the value the less likely will be the associated action relative to other choices.

**blank-page-penalty**
The penalty for having a blank page in the middle of a score. If blank-page-penalty is large and ly:page-turn-breaking is selected, then
LilyPond will be less likely to insert a page in the middle of a score. Instead, it will space out the music further to fill the blank page and the following one. Default: 5.

**blank-last-page-penalty**

The penalty for ending the score on an even-numbered page. If `blank-last-page-penalty` is large and `ly:page-turn-breaking` is selected, then LilyPond will be less likely to produce a score in which the last page is even-numbered. Instead, it will adjust the spacing in order to use one page more or one page less. Default: 0.

**blank-after-score-page-penalty**

The penalty for having a blank page after the end of one score and before the next. By default, this is smaller than `blank-page-penalty`, so that blank pages after scores are inserted in preference to blank pages within a score. Default: 2.

**See also**


Installed Files: ly/paper-defaults-init.ly

\paper variables for page numbering

Default values not listed here are defined in ly/paper-defaults-init.ly

**auto-first-page-number**

The page breaking algorithm is affected by the first page number being odd or even. If set to true, the page breaking algorithm will decide whether to start with an odd or even number. This will result in the first page number remaining as is or being increased by one. Default: #f.

**first-page-number**

The value of the page number on the first page.

**print-first-page-number**

If set to true, a page number is printed on the first page.

**print-page-number**

If set to false, page numbers are not printed.

**page-number-type**

The type of numerals used for page numbers. Choices include `roman-lower`, `roman-upper` and `arabic`. Default: 'arabic.

**See also**

Installed Files: ly/paper-defaults-init.ly

**Known issues and warnings**

Odd page numbers are always on the right. If you want the music to start on page 1 there must be a blank page on the back of the cover page so that page 1 is on the right hand side.

**Miscellaneous \paper variables**

**page-spacing-weight**

The relative importance of page (vertical) spacing and line (horizontal) spacing. High values will make page spacing more important. Default: 10.
print-all-headers
If set to true, this will print all headers for each \score in the output. Normally only the piece and opus header variables are printed. Default: \#f.

system-separator-markup
A markup object that is inserted between systems, often used for orchestral scores. Default: unset. The \slashSeparator markup, defined in ly/titling-init.ly, is provided as a sensible default, for example:

\(\text{(set-default-paper-size "a8")}\)

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = \#f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}

\timesig.C44/clefs.G/noteheads.s0
\noteheads.s0/clefs.G
\noteheads.s0/clefs.G
\timesig.C44/clefs.G
\timesig.C44/clefs.G
\timesig.C44/clefs.G

footnote-separator-markup
A markup object that is inserted above the footnote texts at the bottom of the page. Default: a centered horizontal line, defined in ly/paper-defaults-init.ly.

See also
Snippets: Section “Spacing” in Snippets.

Known issues and warnings
The default page header puts the page number and the instrument field from the \header block on a line.

4.2 Score layout
This section discusses score layout options for the \layout block.
4.2.1 The \layout block

While the \paper block contains settings that relate to the page formatting of the whole document, the \layout block contains settings for score-specific layout. To set score layout options globally, enter them in a toplevel \layout block. To set layout options for an individual score, enter them in a \layout block inside the \score block, after the music. Settings that can appear in a \layout block include:

- the layout-set-staff-size scheme function,
- context modifications in \context blocks, and
- \paper variables that affect score layout.

The layout-set-staff-size function is discussed in the next section, Section 4.2.2 [Setting the staff size], page 559. Context modifications are discussed in a separate chapter; see Section 5.1.4 [Modifying context plug-ins], page 605, and Section 5.1.5 [Changing context default settings], page 607.

The \paper variables that can appear in a \layout block, with default values taken from the \paper block are:

- line-width, ragged-right and ragged-last (see \paper variables for widths and margins], page 551)
- indent and short-indent (see \paper variables for shifts and indents], page 553)
- system-count (see \paper variables for line breaking], page 554)

Here is an example \layout block:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGrouper.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Multiple \layout blocks can be entered as toplevel expressions. This can, for example, be useful if different settings are stored in separate files and included optionally. Internally, when a \layout block is evaluated, a copy of the current \layout configuration is made, then any changes defined within the block are applied and the result is saved as the new current configuration. From the user’s perspective the \layout blocks are combined, but in conflicting situations (when the same property is changed in different blocks) the later definitions take precedence.

For example, if this block:

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```
is placed after the one from the preceding example the 'padding and 'color overrides for TextScript are combined, but the later 'thickness override for Glissando replaces (or hides) the earlier one.

\layout blocks may be assigned to variables for reuse later, but the way this works is slightly but significantly different from writing them literally.

If a variable is defined like this:

```latex
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
  }
}
```

it will hold the current \layout configuration with the NoteHead.font-size override added, but this combination is not saved as the new current configuration. Be aware that the ‘current configuration’ is read when the variable is defined and not when it is used, so the content of the variable is dependent on its position in the source.

The variable can then be used inside another \layout block, for example:

```latex
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

A \layout block containing a variable, as in the example above, does not copy the current configuration but instead uses the content of \layoutVariable as the base configuration for the further additions. This means that any changes defined between the definition and the use of the variable are lost.

If layoutVariable is defined (or \include d) immediately before being used, its content is just the current configuration plus the overrides defined within it. So in the example above showing the use of \layoutVariable the final \layout block would consist of:

```
TextScript.padding = #1
TextScript.color = #magenta
Glissando.thickness = #1.5
NoteHead.font-size = #4
NoteHead.color = #red
```

plus the indent and the StaffGrouper overrides.

But if the variable had already been defined before the first \layout block the current configuration would now contain only

```
NoteHead.font-size = #4 % (written in the variable definition)
NoteHead.color = #red % (added after the use of the variable)
```

If carefully planned, \layout variables can be a valuable tool to structure the layout design of sources, and also to reset the \layout configuration to a known state.

See also

Notation Reference: Section 5.1.5 [Changing context default settings], page 607.
Snippets: Section “Spacing” in Snippets.
4.2.2 Setting the staff size

The default staff size is 20 points, which corresponds to a staff height of 7.03mm (one point is equal to 100/7227 of an inch, or 2540/7227 mm). The staff size may be changed in three ways:

1. To set the staff size globally for all scores in a file (or in a \book block, to be precise), use \set-global-staff-size:

   #(set-global-staff-size 14)

   The above example sets the global default staff size to 14pt (4.92mm) and scales all fonts accordingly.

2. To set the staff size for a single score within a book, use layout-set-staff-size inside that score’s layout block:

   \score {
     ...
     \layout {
       #(layout-set-staff-size 14)
     }
   }

3. To set the staff size for a single staff within a system, use the \magnifyStaff command.

   For example, traditionally engraved chamber music scores with piano often used 7mm piano staves while the other staves were typically between 3/5 and 5/7 as large (between 60% and 71%). To achieve the 5/7 proportion, use:

   \score {
     \new Staff \with {
       \magnifyStaff #5/7
     } { ... }
     \new PianoStaff { ... }
   }

   If you happen to know which fontSize you wish to use, you could use the following form:

   \score {
     \new Staff \with {
       \magnifyStaff #(magstep -3)
     } { ... }
     \new PianoStaff { ... }
   }

   To emulate the look of traditional engraving, it is best to avoid reducing the thickness of the staff lines.

Automatic font weight at different sizes

The Emmentaler font provides the set of Feta musical glyphs in eight different sizes; each one tuned for a different staff size. The smaller the glyph size, the “heavier” it becomes, so as to match the relatively thicker staff lines. Recommended glyphs sizes are listed in the following table:

<table>
<thead>
<tr>
<th>font name</th>
<th>staff height (pt)</th>
<th>staff height (mm)</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>feta11</td>
<td>11.22</td>
<td>3.9</td>
<td>pocket scores</td>
</tr>
<tr>
<td>feta13</td>
<td>12.60</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>feta14</td>
<td>14.14</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4: Spacing issues

<table>
<thead>
<tr>
<th>Font Size</th>
<th>Height (mm)</th>
<th>Width (mm)</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>feta16</td>
<td>15.87</td>
<td>5.6</td>
<td>song books</td>
</tr>
<tr>
<td>feta18</td>
<td>17.82</td>
<td>6.3</td>
<td>standard parts</td>
</tr>
<tr>
<td>feta20</td>
<td>20</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>feta23</td>
<td>22.45</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>feta26</td>
<td>25.2</td>
<td>8.9</td>
<td></td>
</tr>
</tbody>
</table>

See also

Notation Reference: [Selecting notation font size], page 225, Section A.8 [The Emmentaler font], page 683.

Snippets: Section “Spacing” in Snippets.

### 4.3 Breaks

#### 4.3.1 Line breaking

Line breaks are normally determined automatically. They are chosen so that lines look neither cramped nor loose, and consecutive lines have similar density.

To manually force a line break at a bar line, use the \break command:

\relative c' { c4 c c c | \break c4 c c c | }

By default, a \break command inserted in the ‘middle’ of a measure will be ignored (and a warning message will be output during the compilation of the LilyPond file). Adding an invisible bar line – ‘\bar ""’ – before the \break command will force the issue:

\relative c' { c4 c c \bar "" \break c | c4 c c c | }

A \break command that occurs at a bar line will also be ignored if the previous measure ends in the middle of a note (e.g., when a tuplet begins in one measure and ends in another). In this
case remove the `Forbid_line_break_engraver` from the `Voice` context and, use a simultaneous music construction inserting the \break at the appropriate place in the second ‘voice’:

```latex
\new Voice \with {
    \remove "Forbid_line_break_engraver"
} \relative {
    \<<
    \{ c''2. \tuplet 3/2 \{ c4 c c \} c2. | \}
    \{ s1 | \break s1 | \}
    \>>
}
```

Similarly, by default, line breaks are ignored when beams cross bar lines. Use the \override `Beam.breakable = ##t` command to force this:

```latex
\relative c' {
    \override Beam.breakable = ##t
    c2. c8[ c | \break
    c8 c] c2. |
}
```

The \noBreak command will prevent a line break at the bar line where it is inserted.

Within a score, automatic line breaking is prevented within music lying between `\autoLineBreaksOff` and `\autoLineBreaksOn` commands. If automatic page breaks should also be prevented, the commands `\autoBreaksOff` and `\autoBreaksOn` should be used. Manual breaks are unaffected by these commands. Note that inhibiting automatic line breaks may cause music to run over the right margin if it cannot all be contained within one line.

Automatic line breaks (but not page breaks) may be enabled at single bar lines by using `\once \autoLineBreaksOn` at a bar line. This identifies a permitted rather than a forced line break.

The most basic settings influencing line spacing are `indent` and `line-width`. They are set in the `layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `layout` block, then systems end at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is.
The option `ragged-last` is similar to `ragged-right`, but affects only the last line of the piece.

```latex
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

For line breaks at regular intervals use `\break` separated by skips and repeated with `\repeat`. For example, this would cause the following 28 measures (assuming 4/4 time) to be broken every 4 measures, and only there:

```latex
<<
\repeat unfold 7 {
  s1 \noBreak s1 \noBreak
  s1 \noBreak s1 \break
}
{ the actual music... }
>>
```

Predefined commands

- `\break`, `\noBreak`, `\autoBreaksOff`, `\autoBreaksOn`, `\autoLineBreaksOff`, `\autoLineBreaksOn`.

Selected Snippets

**Using an extra voice for breaks**

Often it is easier to manage line and page-breaking information by keeping it separate from the music by introducing an extra voice containing only skips along with the `\break`, `pageBreak` and other layout information.

This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of `NonMusicalPaperColumnGrob`.

```latex
music = \relative c'' { c4 c c c }
```

```latex
\score {
  \new Staff <<
  \new Voice {
    s1 * 2 \break
    s1 * 3 \break
    s1 * 6 \break
    s1 * 5 \break
  }
  \new Voice {
    \repeat unfold 2 { \music }
    \repeat unfold 3 { \music }
    \repeat unfold 6 { \music }
    \repeat unfold 5 { \music }
  }
>>
}
```

```
``
See also

Notation Reference: [paper variables for line breaking], page 554, Section 4.2.1 [The layout block], page 557.
Snippets: Section “Spacing” in Snippets.
Internals Reference: Section “LineBreakEvent” in Internals Reference.

Known issues and warnings

Placing \autoLineBreaksOff or \autoBreaksOff before any music will cause error messages to appear. Always place these commands after some music.

4.3.2 Page breaking

This section describes the different page breaking methods, and how to modify them.

Manual page breaking

The default page breaking may be overridden by inserting \pageBreak or \noPageBreak commands. These commands are analogous to \break and \noBreak. They should be inserted at a bar line. These commands force and forbid a page-break from happening at that bar line. Of course, the \pageBreak command also forces a line break.

The \pageBreak and \noPageBreak commands may also be inserted at top-level, between scores and top-level markups.

Within a score, automatic page breaks are prevented within music lying between \autoPageBreaksOff and \autoPageBreaksOn commands. Manual page breaks are unaffected by these commands.

There are also analogous settings to \ragged-right and \ragged-last which have the same effect on vertical spacing. If \ragged-bottom is set to #t the systems will not be justified vertically. When \ragged-last-bottom is set to #t, as it is by default, empty space is allowed at the bottom of the final page (or the final page in each \bookpart). See Section 4.1.3 [Fixed vertical spacing \paper variables], page 548.

Page breaks are computed by the \page-breaking function. LilyPond provides several algorithms for computing page breaks, including ly:optimal-breaking, ly:page-turn-breaking and ly:minimal-breaking. The default is ly:optimal-breaking, but the value can be changed in the \paper block:

\paper {
  page-breaking = #ly:page-turn-breaking
}
When a book has many scores and pages, the page breaking problem may be difficult to solve, requiring large processing time and memory. To ease the page breaking process, \texttt{bookpart} blocks are used to divide the book into several parts: the page breaking occurs separately on each part. Different page breaking functions may also be used in different book parts.

\begin{verbatim}
\bookpart {
  \header {
    subtitle = "Preface"
  }
  \paper {
    %% In a part consisting mostly of text,
    %% ly:minimal-breaking may be preferred
    page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In this part, consisting of music, the default optimal
  %% page breaking function is used.
  \header {
    subtitle = "First movement"
  }
  \score { ... }
  ...
}
\end{verbatim}

\textbf{Predefined commands}

\texttt{\pageBreak}, \texttt{\noPageBreak}, \texttt{\autoPageBreaksOn}, \texttt{\autoPageBreaksOff}.

\textbf{See also}

Notation Reference: [paper variables for page breaking], page 554.
Snippets: Section “Spacing” in \textit{Snippets}.

\textbf{Known issues and warnings}

The \texttt{\once} prefix is ineffective with \texttt{\autoPageBreaksOn} and \texttt{\autoPageBreaksOff}. If auto page breaking is off and is then turned on to permit a page break, it must remain on for a few bars (the precise number of bars depends on the score) before being turned off, else the opportunity to break the page will not be taken.

\textbf{Optimal page breaking}

The \texttt{ly:optimal-breaking} function is LilyPond’s default method of determining page breaks. It attempts to find a page breaking that minimizes cramping and stretching, both horizontally and vertically. Unlike \texttt{ly:page-turn-breaking}, it has no concept of page turns.

\textbf{See also}

Snippets: Section “Spacing” in \textit{Snippets}.

\textbf{Minimal page breaking}

The \texttt{ly:minimal-breaking} function performs minimal computations to calculate the page breaking: it fills a page with as many systems as possible before moving to the next one.
Thus, it may be preferred for scores with many pages, where the other page breaking functions could be too slow or memory demanding, or a lot of texts. It is enabled using:

\paper {
  page-breaking = #ly:minimal-breaking
}

See also

Snippets: Section “Spacing” in Snippets.

One-page page breaking

The ly:one-page-breaking function is a special-purpose page breaking algorithm that automatically adjusts the page height to fit the music, so that everything fits on a single page. The paper-height variable in the paper block is ignored, but other settings work as usual. In particular, the spacing between the last system (or top level markup) and the footer can be customized with last-bottom-spacing in the paper block. The width of the page is left unmodified by default but can be set with paper-width in the paper block.

Known issues and warnings

ly:one-page-breaking is not currently compatible with \bookpart.

One-line page breaking

The ly:one-line-breaking function is a special-purpose page breaking algorithm that puts each score on its own page, and on a single line. No titles or margins are typeset; only the score is displayed.

The page width is adjusted so that the longest score fits on one line. In particular, paper-width, line-width and indent variables in the \paper block are ignored, although left-margin and right-margin are still honored. The height of the page is left unmodified.

One-line-auto-height page breaking

The ly:one-line-auto-height-breaking function works just like ly:one-line-breaking except the page height is automatically modified to fit the height of the music. Specifically, the paper-height variable in the \paper block is set so that it spans the height of the tallest score plus the top-margin and bottom-margin.

Note that the top-system-spacing setting will affect the vertical position of the music. Set it to ##f in a paper block to simply place the music between the top and bottom margins.

Optimal page turning

Often it is necessary to find a page breaking configuration so that there is a rest at the end of every second page. This way, the musician can turn the page without having to miss notes. The ly:page-turn-breaking function attempts to find a page breaking minimizing cramping and stretching, but with the additional restriction that it is only allowed to introduce page turns in specified places.

There are two steps to using this page breaking function. First, you must enable it in the \paper block, as explained in Section 4.3.2 [Page breaking], page 563. Then you must tell the function where you would like to allow page breaks.

There are two ways to achieve the second step. First, you can specify each potential page turn manually, by inserting \allowPageTurn into your input file at the appropriate places.

If this is too tedious, you can add a Page_turn_engraver to a Staff or Voice context. The Page_turn_engraver will scan the context for sections without notes (note that it does not scan for rests; it scans for the absence of notes. This is so that single-staff polyphony with rests in
one of the parts does not throw off the \texttt{Page\_turn\_engraver}). When it finds a sufficiently long section without notes, the \texttt{Page\_turn\_engraver} will insert an \allowPageTurn at the final bar line in that section, unless there is a ‘special’ bar line (such as a double bar), in which case the \allowPageTurn will be inserted at the final ‘special’ bar line in the section.

The \texttt{Page\_turn\_engraver} reads the context property \texttt{minimumPageTurnLength} to determine how long a note-free section must be before a page turn is considered. The default value for \texttt{minimumPageTurnLength} is (\texttt{ly:make-moment 1/1}). If you want to disable page turns, set it to something ‘very large’.

\begin{verbatim}
\new Staff \with { \consists "Page\_turn\_engraver" } 
{
  a4 b c d |
  R1 | \% a page turn will be allowed here
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
  R1 | \% a page turn will not be allowed here
  a4 b r2 |
  R1\*2 | \% a page turn will be allowed here
  a1
}
\end{verbatim}

When using volta repeats, the \texttt{Page\_turn\_engraver} will only allow a page turn during the repeat if there is enough time at the beginning and end of the repeat to turn the page back. If the repeat is too short then the \texttt{Page\_turn\_engraver} can be used to disable page turns by setting an appropriate value for the context property \texttt{minimumRepeatLengthForPageTurn}. In this case the \texttt{Page\_turn\_engraver} will only allows turns in repeats whose duration is longer than the value specified.

The page turning commands, \pageTurn, \noPageTurn and \allowPageTurn, may also be used at top-level, in top-level markups and between scores.

**Predefined commands**

\pageTurn, \noPageTurn, \allowPageTurn.

**See also**

Notation Reference: [paper variables for line breaking], page 554.
Snippets: Section “Spacing” in Snippets.

**Known issues and warnings**

Use only one \texttt{Page\_turn\_engraver} per score. If there are more, they will interfere with each other.

**See also**

Notation Reference: Section 4.4 [Vertical spacing], page 566.
Snippets: Section “Spacing” in Snippets.

**4.4 Vertical spacing**

Vertical spacing is controlled by three things: the amount of space available (i.e., paper size and margins), the amount of space between systems, and the amount of space between staves inside a system.
4.4.1 Flexible vertical spacing within systems

Three separate mechanisms control the flexible vertical spacing within systems, one for each of the following categories:

- ungrouped staves,
- grouped staves (staves within a staff-group such as ChoirStaff, etc.), and
- non-staff lines (such as Lyrics, ChordNames, etc.).

The height of each system is determined in two steps. First, all of the staves are spaced according to the amount of space available. Then, the non-staff lines are distributed between the staves.

Note that the spacing mechanisms discussed in this section only control the vertical spacing of staves and non-staff lines within individual systems. The vertical spacing between separate systems, scores, markups, and margins is controlled by \texttt{paper} variables, which are discussed in Section 4.1.4 [Flexible vertical spacing \texttt{paper} variables], page 549.

Within-system spacing properties

The within-system vertical spacing mechanisms are controlled by two sets of grob properties. The first set is associated with the \texttt{VerticalAxisGroup} grob, which is created by all staves and non-staff lines. The second set is associated with the \texttt{StaffGrouper} grob, which can be created by staff-groups, but only if explicitly called. These properties are described individually at the end of this section.

The names of these properties (except for \texttt{staff-affinity}) follow the format \texttt{item1-item2-spacing}, where \texttt{item1} and \texttt{item2} are the items to be spaced. Note that \texttt{item2} is not necessarily below \texttt{item1}; for example, \texttt{nonstaff-relatedstaff-spacing} will measure upwards from the non-staff line if \texttt{staff-affinity} is \texttt{UP}.

Each distance is measured between the reference points of the two items. The reference point for a staff is the vertical center of its \texttt{StaffSymbol} (i.e., the middle line if \texttt{line-count} is odd; the middle space if \texttt{line-count} is even). The reference points for individual non-staff lines are given in the following table:

<table>
<thead>
<tr>
<th>Non-staff line</th>
<th>Reference point</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChordNames</td>
<td>baseline</td>
</tr>
<tr>
<td>NoteNames</td>
<td>baseline</td>
</tr>
<tr>
<td>Lyrics</td>
<td>baseline</td>
</tr>
<tr>
<td>Dynamics</td>
<td>mid-height of 'm'</td>
</tr>
<tr>
<td>FiguredBass</td>
<td>highest point</td>
</tr>
<tr>
<td>FretBoards</td>
<td>top line</td>
</tr>
</tbody>
</table>

In the following image, horizontal lines indicate the positions of these reference points:
Each of the vertical spacing grob properties (except `staff-affinity`) uses the same alist structure as the `\paper` spacing variables discussed in Section 4.1.4 [Flexible vertical spacing `\paper` variables], page 549. Specific methods for modifying alists are discussed in Section 5.3.7 [Modifying alists], page 631. Grob properties should be adjusted with an `\override` inside a `\score` or `\layout` block, and not inside a `\paper` block.

The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely re-defines the property:

\begin{verbatim}
\new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
} { ... }
\end{verbatim}

\begin{verbatim}
\new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing = 
        #'((basic-distance . 10)
             (minimum-distance . 9)
             (padding . 1)
             (stretchability . 10))
} { ... }
\end{verbatim}

To change any spacing settings globally, put them in the `\layout` block:

\begin{verbatim}
\layout {
    \context {
        \Staff
            \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
    }
}
\end{verbatim}

Standard settings for the vertical spacing grob properties are listed in Section “VerticalAxisGroup” in Internals Reference and Section “StaffGrouper” in Internals Reference. Default overrides for specific types of non-staff lines are listed in the relevant context descriptions in Section “Contexts” in Internals Reference.

Properties of the VerticalAxisGroup grob

`VerticalAxisGroup` properties are typically adjusted with an `\override` at the `Staff` level (or equivalent).

staff-staff-spacing

Used to determine the distance between the current staff and the staff just below it in the same system, even if one or more non-staff lines (such as Lyrics) are placed between the two staves. Does not apply to the bottom staff of a system.

Initially, the `staff-staff-spacing` of a `VerticalAxisGroup` is a Scheme function that applies the properties of the `StaffGrouper` if the staff is part of a group, or the `default-staff-staff-spacing` of the staff otherwise. This allows staves to be spaced differently when they are grouped. For uniform spacing regardless of grouping, this function may be replaced by a flexible-spacing alist, using the complete-redefinition form of override shown above. If only some values are specified in an override, missing values will be taken from `default-staff-staff-spacing` (if it has values for them).
default-staff-staff-spacing
A flexible-spacing alist defining the staff-staff-spacing used for ungrouped staves, unless staff-staff-spacing has been explicitly set with an \override.

staff-affinity
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. Adjacent non-staff lines should have non-increasing staff-affinity from top to bottom, e.g., a non-staff line set to UP should not immediately follow one that is set to DOWN. Non-staff lines at the top of a system should use DOWN; those at the bottom should use UP. 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. Setting staff-affinity to UP, CENTER, or DOWN causes a staff to be spaced as a non-staff line.

nonstaff-relatedstaff-spacing
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. This means that the placement of a non-staff line depends on both the surrounding staves and the surrounding non-staff lines. Setting the stretchability of one of these types of spacing to a small value will make that spacing dominate. Setting the stretchability to a large value will make that spacing have little effect.

nonstaff-nonstaff-spacing
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.

nonstaff-unrelatedstaff-spacing
The distance between the current non-staff line and the 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. This can be used, for example, to require a minimum amount of padding between a Lyrics line and the staff to which it does not belong.

Properties of the StaffGrouper grob
StaffGrouper properties are typically adjusted with an \override at the StaffGroup level (or equivalent).

staff-staff-spacing
The distance between consecutive staves within the current staff-group. The staff-staff-spacing property of an individual staff’s VerticalAxisGroup grob can be overridden with different spacing settings for that staff.

staffgroup-staff-spacing
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 (such as Lyrics) exist between the two staves. Does not apply to the bottom staff of a system. The staff-staff-spacing property of an individual staff’s VerticalAxisGroup grob can be overridden with different spacing settings for that staff.
See also

Notation Reference: Section 4.1.4 [Flexible vertical spacing \paper variables], page 549, Section 5.3.7 [Modifying alists], page 631.


Internals Reference: Section “Contexts” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference, Section “StaffGrouper” in Internals Reference.

Spacing of ungrouped staves

Staves (such as Staff, DrumStaff, TabStaff, etc.) are contexts that can contain one or more voice contexts, but cannot contain any other staves.

The following properties affect the spacing of ungrouped staves:

- VerticalAxisGroup properties:
  - default-staff-staff-spacing
  - staff-staff-spacing

These grob properties are described individually above; see [Within-system spacing properties], page 567.

Additional properties are involved for staves that are part of a staff-group; see [Spacing of grouped staves], page 571.

The following example shows how the default-staff-staff-spacing property can affect the spacing of ungrouped staves. The same overrides applied to staff-staff-spacing would have the same effect, but would also apply in cases where the staves are combined in a group or groups.

\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing = #'((basic-distance . 8)
                          (minimum-distance . 7)
                          (padding . 1))
  }
}

<<
% The very low note here needs more room than 'basic-distance can provide, so the distance between this staff and the next % is determined by 'padding.
\new Staff { b,2 r | } % Here, 'basic-distance provides enough room, and there is no % need to compress the space (towards 'minimum-distance) to make % room for anything else on the page, so the distance between % this staff and the next is determined by 'basic-distance.
\new Staff { \clef bass g2 r | } % By setting 'padding to a negative value, staves can be made to % collide. The lowest acceptable value for 'basic-distance is 0.
\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing = #'((basic-distance . 3.5)
Spacing issues

Spacing of grouped staves

In orchestral and other large scores, it is common to place staves in groups. The space between groups is typically larger than the space between staves of the same group.

Staff-groups (such as StaffGroup, ChoirStaff, etc.) are contexts that can contain one or more staves simultaneously.

The following properties affect the spacing of staves inside staff-groups:

- VerticalAxisGroup properties:
  - staff-staff-spacing
- StaffGrouper properties:
  - staff-staff-spacing
  - staffgroup-staff-spacing

These grob properties are described individually above; see [Within-system spacing properties], page 567.

The following example shows how properties of the StaffGrouper grob can affect the spacing of grouped staves:

\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = #0
    \override StaffGrouper.staff-staff-spacing.basic-distance = #1
  }
}

\new PianoStaff \with {
  \override StaffGrouper.staffgroup-staff-spacing.basic-distance = #20
} \new Staff { c'1 }

See also

Installed Files: scm/define-grobs.scm.
Snippets: Section “Spacing” in Snippets.
Internals Reference: Section “VerticalAxisGroup” in Internals Reference.
Spacing of non-staff lines

Non-staff lines (such as Lyrics, ChordNames, etc.) are contexts whose layout objects are engraved like staves (i.e., in horizontal lines within systems). Specifically, non-staff lines are non-staff contexts that contain the Section “Axis group engraver” in Internals Reference.

The following properties affect the spacing of non-staff lines:

- **VerticalAxisGroup** properties:
  - staff-affinity
  - nonstaff-relatedstaff-spacing
  - nonstaff-nonstaff-spacing
  - nonstaff-unrelatedstaff-spacing

These grob properties are described individually above; see [Within-system spacing properties], page 567.

The following example shows how the nonstaff-nonstaff-spacing property can affect the spacing of consecutive non-staff lines. Here, by setting the stretchability key to a very high value, the lyrics are able to stretch much more than usual:

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup.nonstaff-nonstaff-spacing.stretchability = #1000
  }
}
```
4.4.2 Explicit staff and system positioning

One way to understand the flexible vertical spacing mechanisms explained above is as a collection of settings that control the amount of vertical padding between staves and systems.

It is possible to approach vertical spacing in a different way using NonMusicalPaperColumn.line-break-system-details. While the flexible vertical spacing mechanisms specify vertical padding, NonMusicalPaperColumn.line-break-system-details can specify exact vertical positions on the page.

NonMusicalPaperColumn.line-break-system-details accepts an associative list of four different settings:

- X-offset
- Y-offset

See also

Snippets: Section “Spacing” in Snippets.
Internals Reference: Section “Contexts” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference.
Chapter 4: Spacing issues

- extra-offset
- alignment-distances

Grob overrides, including the overrides for NonMusicalPaperColumn below, can occur in any of three different places in an input file:
- in the middle of note entry directly
- in a \context block
- in the \with block

When we override NonMusicalPaperColumn, we use the usual \override command in \context blocks and in the \with block. On the other hand, when we override NonMusicalPaperColumn in the middle of note entry, use the special \overrideProperty command. Here are some example NonMusicalPaperColumn overrides with the special \overrideProperty command:

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details #'((X-offset . 20))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 40))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details #'((X-offset . 20) (Y-offset . 40))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details #'((alignment-distances . (15))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details #'((X-offset . 20) (Y-offset . 40) (alignment-distances . (15))
```

To understand how each of these different settings work, we begin by looking at an example that includes no overrides at all.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
    >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
```

This score isolates both line-breaking and page-breaking information in a dedicated voice. This technique of creating a breaks voice will help keep layout separate from music entry as our example becomes more complicated. Also see Section 4.3 [Breaks], page 560.

By using explicit \break commands, the music is divided into five measures per line. Vertical spacing is from LilyPond’s own defaults but the vertical startpoint of each system is set explicitly using the Y-offset pair in the line-break-system-details attribute of the NonMusicalPaperColumn grob:

\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
      \new Voice {
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 0))
        s1*5 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 40))
        s1*5 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 60))
        s1*5 \break
      }
      \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
    >>
    \new Staff {
      \repeat unfold 15 { d'4 d' d' d' }
    }
  >>
}
Note that `line-break-system-details` takes an associative list of potentially many values, but that we set only one value here. Note, too, that the `Y-offset` property here determines the exact vertical position on the page at which each new system will render.

In contrast to the absolute positioning available through `Y-offset` and `X-offset`, relative positioning is possible with the `extra-offset` property of `line-break-system-details`. Placement is relative to the default layout or to the absolute positioning created by setting `X-offset` and `Y-offset`. The property `extra-offset` accepts a pair consisting of displacements along the X-axis and Y-axis.
Now that we have set the vertical startpoint of each system explicitly, we can also set the vertical distances between staves within each system manually. We do this using the `alignment-distances` subproperty of `line-break-system-details`.

\new Staff {
    \repeat unfold 15 { d'4 d' d' d' }
}
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
    \score {
        <<
            \new Staff {
                \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 20)
                    (alignment-distances . (10)))
            s1\*5 \break
            \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 60)
                    (alignment-distances . (15)))
            s1\*5 \break
            \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 85)
                    (alignment-distances . (20)))
            s1\*5 \break
        }
    }
Note that here we assign two different values to the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob. Though the `line-break-system-details` attribute alist accepts many additional spacing parameters (including, for example, a corresponding X-offset pair), we need only set the Y-offset and `alignment-distances` pairs to control the vertical startpoint of every system and every staff. Finally, note that `alignment-distances` specifies the vertical positioning of staves but not of staff groups.

\header { tagline = ##f } \\
\paper { left-margin = 0\mm } \\
\book { \\
  \score { \\
    << \\
    \new Staff << \\
    \new Voice {

\overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 0)
   (alignment-distances . (30 10)))
s1*5 \break
\overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 60)
   (alignment-distances . (10 10)))
s1*5 \break
\overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 100)
   (alignment-distances . (10 30)))
s1*5 \break
\new Voice { \repeat unfold 15 { c'4 c' c' c' } }
>>
\new StaffGroup <<
\new Staff { \repeat unfold 15 { d'4 d' d' d' } }
\new Staff { \repeat unfold 15 { e'4 e' e' e' } }
>>
>>
}
Some points to consider:

- When using `alignment-distances`, lyrics and other non-staff lines do not count as a staff.
- The units of the numbers passed to `X-offset`, `Y-offset`, `extra-offset` and `alignment-distances` are interpreted as multiples of the distance between adjacent staff lines. Positive values move staves and lyrics up, negative values move staves and lyrics down.
- Because the `NonMusicalPaperColumn.line-break-system-details` settings given here allow the positioning of staves and systems anywhere on the page, it is possible to violate
paper or margin boundaries or even to print staves or systems on top of one another. Reasonable values passed to these different settings will avoid this.

See also

Snippets: Section “Spacing” in Snippets.

4.4.3 Vertical collision avoidance

Intuitively, there are some objects in musical notation that belong to the staff and there are other objects that should be placed outside the staff. Objects belonging outside the staff include things such as rehearsal marks, text and dynamic markings (from now on, these will be called outside-staff objects). LilyPond’s rule for the vertical placement of outside-staff objects is to place them as close to the staff as possible but not so close that they collide with another object.

LilyPond uses the outside-staff-priority property to determine whether a grob is an outside-staff object: if outside-staff-priority is a number, the grob is an outside-staff object. In addition, outside-staff-priority tells LilyPond in which order the objects should be placed.

First, LilyPond places all the objects that do not belong outside the staff. Then it sorts the outside-staff objects according to their outside-staff-priority (in increasing order). One by one, LilyPond takes the outside-staff objects and places them so that they do not collide with any objects that have already been placed. That is, if two outside-staff grobs are competing for the same space, the one with the lower outside-staff-priority will be placed closer to the staff.

A listing of outside-staff-priorities may be found in Section “The outside-staff-priority property” in Learning Manual.

```
\relative c'' {
  \once \override TextScript.outside-staff-priority = #1
  c4_"Text"\pp r2.
  \once \override TextScript.outside-staff-priority = #1
  c4_"Text"\pp % this time the text will be closer to the staff
  r2.
  % by setting outside-staff-priority to a non-number,
  % we disable the automatic collision avoidance
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Text"\pp % now they will collide
}
```

The vertical padding around outside-staff objects can be controlled with outside-staff-padding.

```
\relative {
  \once \override TextScript.outside-staff-padding = #0
  a'4-"outside-staff-padding = #0"
  \once \override TextScript.outside-staff-padding = #3
  d-"outside-staff-padding = #3"
  c-"default outside-staff-padding"
}
```
By default, outside-staff objects are placed so they avoid a horizontal collision with previously-positioned grobs. This can lead to situations in which objects are placed close to each other horizontally. As shown in the example below, setting `outside-staff-horizontal-padding` increases the horizontal spacing required, and in this case moves the text up to prevent it from getting too close to the ledger lines.

\relative { 
  c'4"Word" c c'2 
  R1 
  \once \override TextScript.outside-staff-horizontal-padding = #1 
  c,,4"Word" c c'2 
} 

See also
Snippets: Section “Spacing” in Snippets.

4.5 Horizontal spacing

4.5.1 Horizontal spacing overview

The spacing engine translates differences in durations into stretchable distances (‘springs’) of differing lengths. Longer durations get more space, shorter durations get less. The shortest durations get a fixed amount of space (which is controlled by `shortest-duration-space` in the Section “SpacingSpanner” in Internals Reference object). The longer the duration, the more space it gets: doubling a duration adds `spacing-increment` of space to the note.

For example, the following piece contains lots of half, quarter, and 8th notes; the eighth note is followed by 1 note head width (NHW). The quarter note is followed by 2 NHW, the half by 3 NHW, etc.

\relative c' { 
  c2 c4. c8 
  c4. c8 c4. c8 
  c8 c c4 c c 
}
Normally, \textit{spacing-increment} is set to 1.2 staff space, which is approximately the width of a note head, and \textit{shortest-duration-space} is set to 2.0, meaning that the shortest note gets 2.4 staff space (2.0 times the \textit{spacing-increment}) of horizontal space. This space is counted from the left edge of the symbol, so the shortest notes are generally followed by one NHW of space.

If one would follow the above procedure exactly, then adding a single 32nd note to a score that uses 8th and 16th notes, would widen up the entire score a lot. The shortest note is no longer a 16th, but a 32nd, thus adding 1 NHW to every note. To prevent this, the shortest duration for spacing is not the shortest note in the score, but rather the one which occurs most frequently.

The most common shortest duration is determined as follows: in every measure, the shortest duration is determined. The most common shortest duration is taken as the basis for the spacing, with the stipulation that this shortest duration should always be equal to or shorter than an 8th note.

These durations may also be customized. If you set the \textit{common-shortest-duration} in Section “SpacingSpanner” in \textit{Internals Reference}, then this sets the base duration for spacing. The maximum duration for this base (normally an 8th), is set through \textit{base-shortest-duration}.

Notes that are even shorter than the common shortest note are followed by a space that is proportional to their duration relative to the common shortest note. So if we were to add only a few 16th notes to the example above, they would be followed by half a NHW:

\begin{verbatim}
\relative { c'\textquoteright2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
\end{verbatim}

As explained in the \textit{Essay on automated music engraving}, stem directions will influence spacing (see Section “Optical spacing” in \textit{Essay}) and can be adjusted using the \textit{stem-spacing-correction} property of the Section “NoteSpacing” in \textit{Internals Reference} object (which are generated for every Section “Voice” in \textit{Internals Reference} context).

The \textit{StaffSpacing} object (generated in Section “Staff” in \textit{Internals Reference} context) contains the same property for controlling the stem/bar line spacing.

The following example shows this; once with the default settings and once with an exaggerated adjustment:

\begin{verbatim}
\relative { c\textquoteright\textquoteright2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
\end{verbatim}

Proportional notation is supported; see Section 4.5.5 [Proportional notation], page 587.

\textbf{See also}

- \textit{Essay} on automated music engraving: Section “Optical spacing” in \textit{Essay}.
- \textit{Snippets}: Section “Spacing” in \textit{Snippets}.
Known issues and warnings

There is no convenient mechanism to manually override spacing. The following work-around may be used to insert extra space into a score, adjusting the padding value as necessary.

\override Score.NonMusicalPaperColumn.padding = #10

No work-around exists for decreasing the amount of space.

4.5.2 New spacing section

New sections with different spacing parameters can be started with the \newSpacingSection command. This is useful for sections with different notions of ‘long’ and ‘short’ notes. The \newSpacingSection command creates a new SpacingSpanner object at that musical moment.

In the following example the time signature change introduces a new section, and the 16ths notes are automatically spaced slightly wider apart.

\relative c' {
  \time 2/4
  c4 c8 c
  c8 c c16[ c c8] c4
  \newSpacingSection
  \time 4/16
  c16[ c c8]
}

If the automatic spacing adjustments do not give the required spacing, manual \overrides may be applied to its properties. These must be applied at the same musical moment as the \newSpacingSection command itself and will then affect the spacing of all the following music until the properties are changed in a new spacing section, for example:

\relative c' {
  \time 4/16
  c16[ c c8]
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #2
  c16[ c c8]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
  c16[ c c8]
}

See also

Snippets: Section “Spacing” in Snippets.
Internals Reference: Section “SpacingSpanner” in Internals Reference.
### 4.5.3 Changing horizontal spacing

Horizontal spacing may be altered with the `base-shortest-duration` property. Here we compare the same music; once without altering the property, and then altered. Larger values of `ly:make-moment` will produce smaller music. Note that `ly:make-moment` constructs a duration, so 1/4 is a longer duration than 1/16.

\score {
\relative {
  g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
  g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
  g4 e e2 | f4 d d2 | c4 e g g | c,1 |
}
}

\score {
\relative {
  g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
  g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
  g4 e e2 | f4 d d2 | c4 e g g | c,1 |
}
\layout {
\context {
  \Score
  \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
}
}\context {
\Score
\override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
}\context {
\Score
\override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
}
Uniform stretching of tuplets

By default, spacing in tuplets depends on various non-duration factors (such as accidentals, clef changes, etc). To disregard such symbols and force uniform equal-duration spacing, use `Score.SpacingSpanner.uniform-stretching`. This property can only be changed at the beginning of a score,

\score { << \new Staff \relative c' { \tuplet 5/4 { c8 c c c c } c8 c c c } \new Staff \relative c' { c8 c c \tuplet 5/4 { c8 c c c c } } >> \layout { \context { \Score \override SpacingSpanner.uniform-stretching = ##t } } \}}

Strict note spacing

When `strict-note-spacing` is set, notes are spaced without regard for clefs, bar lines, and grace notes,

\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative { c'8[ c \clef alto c \grace { c16 c } c8 c c ] c32[ c ] }
4.5.4 Line width

The most basic settings influencing the spacing are \texttt{indent} and \texttt{line-width}. They are set in the \texttt{\layout} block. They control the indentation of the first line of music, and the lengths of the lines.

If \texttt{ragged-right} is set to \texttt{true} in the \texttt{\layout} block, then systems ends at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is. The normal default setting is false, but if the score has only one system the default value is true.

The option \texttt{ragged-last} is similar to \texttt{ragged-right}, but only affects the last line of the piece. No restrictions are put on that line. The result is similar to formatting text paragraphs. In a paragraph, the last line simply takes its natural horizontal length.

\begin{verbatim}
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
\end{verbatim}

See also

Snippets: Section “Spacing” in Snippets.

4.5.5 Proportional notation

LilyPond supports proportional notation, a type of horizontal spacing in which each note consumes an amount of horizontal space exactly equivalent to its rhythmic duration. This type of proportional spacing is comparable to horizontal spacing on top of graph paper. Some late 20th- and early 21st-century scores use proportional notation to clarify complex rhythmic relationships or to facilitate the placement of timelines or other graphics directly in the score.

LilyPond supports five different settings for proportional notation, which may be used together or alone:

- \texttt{proportionalNotationDuration}
- \texttt{uniform-stretching}
- \texttt{strict-note-spacing}
- \texttt{\remove "Separating_line_group_engraver"}
- \texttt{\override PaperColumn.used = ##t}

In the examples that follow, we explore these five different proportional notation settings and examine how these settings interact.

We start with the following one-measure example, which uses classical spacing with \texttt{ragged-right} turned on.

\begin{verbatim}
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
}
\end{verbatim}
Notice that the half note which begins the measure takes up far less than half of the horizontal space of the measure. Likewise, the sixteenth notes and sixteenth-note quintuplets (or twentieth notes) which end the measure together take up far more than half the horizontal space of the measure.

In classical engraving, this spacing may be exactly what we want because we can borrow horizontal space from the half note and conserve horizontal space across the measure as a whole.

On the other hand, if we want to insert a measured timeline or other graphic above or below our score, we need proportional notation. We turn proportional notation on with the proportionalNotationDuration setting.

\begin{verbatim}
\score {
  \context {\Score
    proportionalNotationDuration = #(ly:make-moment 1/20)
  }
}
\end{verbatim}

The half note at the beginning of the measure and the faster notes in the second half of the measure now occupy equal amounts of horizontal space. We could place a measured timeline or graphic above or below this example.

The proportionalNotationDuration setting is a context setting that lives in Score. Remember that context settings can appear in one of three locations within our input file – in a with block, in a context block, or directly in music entry preceded by the set command. As with all context settings, users can pick which of the three different locations they would like to set proportionalNotationDuration in to.

The proportionalNotationDuration setting takes a single argument, which is the reference duration against that all music will be spaced. The LilyPond Scheme function make-moment takes two arguments – a numerator and denominator which together express some fraction of a whole note. The call (ly:make-moment 1/20) therefore produces a reference duration of a twentieth note. Values such as (ly:make-moment 1/16), (ly:make-moment 1/8), and (ly:make-moment 3/97) are all possible as well.

How do we select the right reference duration to pass to proportionalNotationDuration? Usually by a process of trial and error, beginning with a duration close to the fastest (or smallest) duration in the piece. Smaller reference durations space music loosely; larger reference durations space music tightly.

\begin{verbatim}
\score {
  <<
    \new RhythmicStaff { c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
>>
\end{verbatim}
Note that too large a reference duration – such as the eighth note, above – spaces music too tightly and can cause note head collisions. Also that proportional notation in general takes up...
more horizontal space than classical spacing. Proportional spacing provides rhythmic clarity at the expense of horizontal space.

Next we examine how to optimally space overlapping tuplets.

We start by examining what happens to our original example, with classical spacing, when we add a second staff with a different type of tuplet.

\score {
  \new RhythmicStaff {
    \c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
  \new RhythmicStaff {
    \tuplet 9/8 { \c8 8 8 8 8 8 8 8 8 }
  }
}
But if we look very carefully we can see that notes of the second half of the 9-tuplet space ever so slightly more widely than the notes of the first half of the 9-tuplet. To ensure uniform stretching, we turn on `uniform-stretching`, which is a property of `SpacingSpanner`.

\score {
    \new RhythmicStaff {
        c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
        \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
}

Our two-staff example now spaces exactly, our rhythmic relationships are visually clear, and we can include a measured timeline or graphic if we want.

Note that the LilyPond’s proportional notation package expects that all proportional scores set the `SpacingSpanner`’s `uniform-stretching` attribute to `##t`. Setting `proportionalNotationDuration` without also setting the `SpacingSpanner`’s `uniform-stretching` attribute to `##t` will, for example, cause Skips to consume an incorrect amount of horizontal space.

The `SpacingSpanner` is an abstract grob that lives in the `Score` context. As with our settings of `proportionalNotationDuration`, overrides to the `SpacingSpanner` can occur in any of three different places in our input file – in the `Score` \with block, in a `Score` \context block, or in note entry directly.

There is by default only one `SpacingSpanner` per `Score`. This means that, by default, `uniform-stretching` is either turned on for the entire score or turned off for the entire score. We can, however, override this behavior and turn on different spacing features at different places in the score. We do this with the command `\newSpacingSection`. See Section 4.5.2 [New spacing section], page 584, for more info.

Next we examine the effects of the `Separating_line_group_engraver` and see why proportional scores frequently remove this engraver. The following example shows that there is a small amount of “prefatory” space just before the first note in each system.

\paper {
    indent = #0
}

\new Staff {


The amount of this prefatory space is the same whether after a time signature, a key signature or a clef. \texttt{Separating\_line\_group\_engraver} is responsible for this space. Removing \texttt{Separating\_line\_group\_engraver} reduces this space to zero.

non-musical elements like time signatures, key signatures, clefs and accidentals are problematic in proportional notation. None of these elements has rhythmic duration. But all of these elements consume horizontal space. Different proportional scores approach these problems differently.

It may be possible to avoid spacing problems with key signatures simply by not having any. This is a valid option since most proportional scores are contemporary music. The same may be true of time signatures, especially for those scores that include a measured timeline or other graphic. But these scores are exceptional and most proportional scores include at least some time signatures. Clefs and accidentals are even more essential.

So what strategies exist for spacing non-musical elements in a proportional context? One good option is the \texttt{strict\_note\_spacing} property of \texttt{SpacingSpanner}. Compare the two scores below:

\begin{verbatim}
\new Staff {
    \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
    c'1 \break c'1
}\end{verbatim}
Both scores are proportional, but the spacing in the first score is too loose because of the clef change. The spacing of the second score remains strict, however, because strict-note-spacing is turned on. Turning on strict-note-spacing causes the width of time signatures, key signatures, clefs and accidentals to play no part in the spacing algorithm.

In addition to the settings given here, there are other settings that frequently appear in proportional scores. These include:

- \override SpacingSpanner.strict-grace-spacing = ##t
- \set tupletFullLength = ##t
- \override Beam.breakable = ##t
- \override Glissando.breakable = ##t
- \override TextSpanner.breakable = ##t
- \remove "Forbid_line_break_engraver" in the Voice context

These settings space grace notes strictly, extend tuplet brackets to mark both rhythmic start- and stop-points, and allow spanning elements to break across systems and pages. See the respective parts of the manual for these related settings.

See also

Notation Reference: Section 4.5.2 [New spacing section], page 584.

Snippets: Section “Spacing” in Snippets.

4.6 Fitting music onto fewer pages

Sometimes you can end up with one or two staves on a second (or third, or fourth... ) page. This is annoying, especially if you look at previous pages and it looks like there is plenty of room left on those.

When investigating layout issues, annotate-spacing is an invaluable tool. This command prints the values of various layout spacing variables; for more details see the following section, Section 4.6.1 [Displaying spacing], page 594.
4.6.1 Displaying spacing

To graphically display the dimensions of vertical layout variables that may be altered for page formatting, set `annotate-spacing` in the `\paper` block:

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```

All layout dimensions are displayed in staff-spaces, regardless of the units specified in the `\paper` or `\layout` block. In the above example, `paper-height` has a value of 59.75 staff-spaces, and the `staff-size` is 20 points (the default value). Note that:

1 point = \((25.4/72.27)\) mm

1 staff-space = \((\text{staff-size})/4\) pts
= \((\text{staff-size})/4 \times (25.4/72.27)\) mm

In this case, one staff-space is approximately equal to 1.757mm. Thus the `paper-height` measurement of 59.75 staff-spaces is equivalent to 105 millimeters, the height of A6 paper in landscape orientation. The pairs \((a,b)\) are intervals, where \(a\) is the lower edge and \(b\) the upper edge of the interval.

**See also**

Notation Reference: Section 4.2.2 [Setting the staff size], page 559.

Snippets: Section “Spacing” in Snippets.
4.6.2 Changing spacing

The output of annotate-spacing reveals vertical dimensions in great detail. For details about modifying margins and other layout variables, see Section 4.1 [Page layout], page 546.

Other than margins, there are a few other options to save space:

- Force systems to move as close together as possible (to fit as many systems as possible onto a page) while being spaced so that there is no blank space at the bottom of the page.

\[
\text{\textbackslash{}paper \{} \\
\text{system-system-spacing} = ' ((basic-distance . 0.1) (padding . 0)) \\
\text{ragged-last-bottom} = ##f \\
\text{ragged-bottom} = ##f \\
\text{\}}
\]

- Force the number of systems. This can help in two ways. Just setting a value, even the same value as the number of systems being typeset by default, will sometimes cause more systems to be fitted onto each page, as an estimation step is then bypassed, giving a more accurate fit to each page. Also, forcing an actual reduction in the number of systems may save a further page. For example, if the default layout has 11 systems, the following assignment will force a layout with 10 systems.

\[
\text{\textbackslash{}paper \{} \\
\text{system-count} = #10 \\
\text{\}}
\]

- Force the number of pages. For example, the following assignment will force a layout with 2 pages.

\[
\text{\textbackslash{}paper \{} \\
\text{page-count} = #2 \\
\text{\}}
\]

- Avoid (or reduce) objects that increase the vertical size of a system. For example, volta brackets for alternative repeat endings require extra space. If these endings are spread over two systems, they take up more space than if they were on the same system. As another example, dynamics that ‘stick out’ of a system can be moved closer to the staff:

\[
\text{\textbackslash{}relative e' \{} \\
\text{e4 c g\f c} \\
\text{e4 c g-\textbackslash{}tweak X-offset #-2.7 \f c} \\
\text{\}}
\]

- Alter the horizontal spacing via SpacingSpanner. For more details, see Section 4.5.3 [Changing horizontal spacing], page 585. The following example illustrates the default spacing:

\[
\text{\textbackslash{}score \{} \\
\text{\relative \{} \\
\text{g'4 e e2 |} \\
\text{f4 d d2 |} \\
\text{c4 d e f |} \\
\text{g4 g g2 |} \\
\text{g4 e e2 |}
\text{\}}
\]
The next example modifies common-shortest-duration from a value of 1/4 to 1/2. The quarter note is the most common and shortest duration in this example, so by making this duration longer, a ‘squeezing’ effect occurs:

```
\score {
  \relative {
    g'4 e e2 | 
    f4 d d2 | 
    c4 d e f | 
    g4 g g2 | 
    g4 e e2 | 
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration = #(ly:make-moment 1/2)
    }
  }
}
```

The common-shortest-duration property cannot be modified dynamically, so it must always be placed in a \context block so that it applies to the whole score.

**See also**

Notation Reference: Section 4.1 [Page layout], page 546, Section 4.5.3 [Changing horizontal spacing], page 585.

Snippets: Section “Spacing” in Snippets.
5 Changing defaults

The purpose of LilyPond’s design is to provide the finest quality output by default. Nevertheless, it may happen that you need to change this default layout. The layout is controlled through a large number of ‘knobs and switches’ collectively called ‘properties’. A tutorial introduction to accessing and modifying these properties can be found in the Learning Manual, see Section “Tweaking output” in Learning Manual. This should be read first. This chapter covers similar ground, but in a style more appropriate to a reference manual.

The definitive description of the controls available for tuning can be found in a separate document: Section “the Internals Reference” in Internals Reference. That manual lists all the variables, functions and options available in LilyPond. It is written as a HTML document, which is available on-line (http://lilypond.org/doc/stable/Documentation/internals/), and is also included with the LilyPond documentation package.

Internally, LilyPond uses Scheme (a LISP dialect) to provide infrastructure. Overriding layout decisions in effect accesses the program internals, which requires Scheme input. Scheme elements are introduced in a .ly file with the hash mark #.\footnote{Section “Scheme tutorial” in Extending, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.}

5.1 Interpretation contexts

This section describes what contexts are, and how to modify them.

See also

Snippets: Section “Contexts and engravers” in Snippets.
Internals Reference: Section “Contexts” in Internals Reference, Section “Engravers and Performers” in Internals Reference.

5.1.1 Contexts explained

Contexts are arranged hierarchically:

Output definitions - blueprints for contexts

This section explains the relevance of output definitions when working with contexts. Examples for actual output definitions are given later (see [Changing all contexts of the same type], page 607).

While music written in a file may refer to context types and names, contexts are created only when the music is actually being interpreted. LilyPond interprets music under control of an ‘output definition’ and may do so for several different output definitions, resulting in different output. The output definition relevant for printing music is specified using \layout.

A much simpler output definition used for producing Midi output is specified using \midi. Several other output definitions are used by LilyPond internally, like when using the part combiner ([Automatic part combining], page 183) or creating music quotes ([Quoting other voices], page 214).

Output definitions define the relation between contexts as well as their respective default settings. While most changes will usually be made inside of a \layout block, Midi-related settings will only have an effect when made within a \midi block.

Some settings affect several outputs: for example, if autoBeaming is turned off in one context, beams count as melismata for the purpose of matching music to lyrics as described in
[Automatic syllable durations], page 278. This matching is done both for printed output as well as for Midi. If changes made to autoBeaming within a context definition of a \layout block are not repeated in the corresponding \midi block, lyrics and music will get out of sync in Midi.

See also


Score - the master of all contexts

This is the top level notation context. No other context can contain a Score context. By default the Score context handles the administration of time signatures and makes sure that items such as clefs, time signatures, and key-signatures are aligned across staves.

A Score context is instantiated implicitly when a \score {...} block is processed.

Top-level contexts - staff containers

StaffGroup

Groups staves while adding a bracket on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically. StaffGroup only consists of a collection of staves, with a bracket in front and spanning bar lines.

ChoirStaff

Identical to StaffGroup except that the bar lines of the contained staves are not connected vertically.

GrandStaff

A group of staves, with a brace on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically.

PianoStaff

Just like GrandStaff, but with support for instrument names to the left of each system.

Intermediate-level contexts - staves

Staff

Handles clefs, bar lines, keys, accidentals. It can contain Voice contexts.

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.

TabStaff

Context for generating tablature. By default lays the music expression out as a guitar tablature, printed on six lines.

DrumStaff

Handles typesetting for percussion. Can contain DrumVoice.

VaticanaStaff

Same as Staff, except that it is designed for typesetting a piece in gregorian style.

MensuralStaff

Same as Staff, except that it is designed for typesetting a piece in mensural style.
Bottom-level contexts - voices

Voice-level contexts initialise certain properties and start appropriate engravers. A bottom-level context is one without defaultchild. While it is possible to let it accept/contain subcontexts, they can only be created and entered explicitly.

Voice
Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and sub-scripts, slurs, ties, and rests. You have to instantiate this explicitly if you require multiple voices on the same staff.

VaticanaVoice
Same as Voice, except that it is designed for typesetting a piece in gregorian style.

MensuralVoice
Same as Voice, with modifications for typesetting a piece in mensural style.

Lyrics
Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

DrumVoice
The voice context used in a percussion staff.

FiguredBass
The context in which BassFigure objects are created from input entered in \figuremode mode.

TabVoice
The voice context used within a TabStaff context. Usually left to be created implicitly.

CueVoice
A voice context used to render notes of a reduced size, intended primarily for adding cue notes to a staff, see [Formatting cue notes], page 217. Usually left to be created implicitly.

ChordNames
Typesets chord names.

5.1.2 Creating and referencing contexts

LilyPond will create lower-level contexts automatically if a music expression is encountered before a suitable context exists, but this is usually successful only for simple scores or music fragments like the ones in the documentation. For more complex scores it is advisable to specify all contexts explicitly with either the \new or \context command. The syntax of these two commands is very similar:

[\new | \context] Context [ = name] [music-expression]

where either \new or \context may be specified. Context is the type of context which is to be created, name is an optional name to be given to the particular context being created and music-expression is a single music expression that is to be interpreted by the engravers and performers in this context.

The \new prefix without a name is commonly used to create scores with many staves:

<<
\new Staff \relative {  
  % leave the Voice context to be created implicitly
  c'4 c  
}
\new Staff \relative {  
  d'4 d  
>>
and to place several voices into one staff:

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    c'8 c c4 c c
  }
  \new Voice \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```

\new should always be used to specify unnamed contexts.

The difference between \new and \context is in the action taken:

- \new with or without a name will always create a fresh, distinct, context, even if one with the same name already exists:

```
\new Staff <<
  \new Voice = "A" \relative {
    \voiceOne
    c'8 c c4 c c
  }
  \new Voice = "A" \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```

- \context with a name specified will create a distinct context only if a context of the same type with the same name in the same context hierarchy does not already exist. Otherwise it will be taken as a reference to that previously created context, and its music expression will be passed to that context for interpretation.

Named contexts may be useful in special cases such as lyrics or figured bass, as demonstrated in [Working with lyrics and variables], page 287, and Section “Vocal ensembles templates”
in *Learning Manual* for the former and [Displaying figured bass], page 441, for the latter. More generally, one application of named contexts is in separating the score layout from the musical content. Either of these two forms is valid:

\begin{verbatim}
\score {
  \% score layout
  \new Staff <<
    \new Voice = "one" {
      \voiceOne
    }
    \new Voice = "two" {
      \voiceTwo
    }
  >>

  \% musical content
  \context Voice = "one" {
    \relative {
      c''4 c c c
    }
  }
  \context Voice = "two" {
    \relative {
      g'8 g g4 g g
    }
  }
  >>
}
\end{verbatim}

\begin{music}
\score {
  \% score layout
  \new Staff <<
    \context Voice = "one" {
      \voiceOne
    }
    \context Voice = "two" {
      \voiceTwo
    }
  >>

  \% musical content
  \context Voice = "one" {
    \relative {
      c''4 c c c
    }
  }
  \context Voice = "two" {

\end{music}
Alternatively, variables may be employed to similar effect. See Section “Organizing pieces with variables” in Learning Manual.

- \context with no name will match the first of any previously created contexts of the same type in the same context hierarchy, even one that has been given a name, and its music expression will be passed to that context for interpretation. This form is rarely useful. However, \context with no name and no music expression is used to set the context in which a Scheme procedure specified with \applyContext is executed:

```
\new Staff \relative { 
  \context Timing 
  \applyContext #(lambda (ctx)
    (newline)
    (display (ly:context-current-moment ctx)))
  c1
}
```

A context must be named if it is to be referenced later, for example when lyrics are associated with music:

```
\new Voice = "tenor" music
...
\new Lyrics \lyricsto "tenor" lyrics
```

For details of associating lyrics with music see [Automatic syllable durations], page 278.

The properties of all contexts of a particular type can be modified in a \layout block (with a different syntax), see [Changing all contexts of the same type], page 607. This construct also provides a means of keeping layout instructions separate from the musical content. If a single context is to be modified, a \with block must be used, see [Changing just one specific context], page 610.

See also


Notation Reference: [Changing just one specific context], page 610, [Automatic syllable durations], page 278.

5.1.3 Keeping contexts alive

Contexts are usually terminated at the first musical moment in which they have nothing to do. So Voice contexts die as soon as they contain no events, Staff contexts die as soon as all the Voice contexts within them contain no events, etc. This can cause difficulties if earlier contexts which have died have to be referenced, for example, when changing staves with \change commands, associating lyrics with a voice with \lyricsto commands, or when adding further musical events to an earlier context.
There is an exception to this general rule: inside of an {...} construct (sequential music), the construct’s notion of the “current context” will descend whenever an element of the sequence ends in a subcontext of the previous current context. This avoids spurious creation of implicit contexts in a number of situations but means that the first context descended into will be kept alive until the end of the expression.

In contrast, the contexts of a <<...>> construct’s (simultaneous music) expression are not carried forth, so enclosing a context creating command in an extra pair of <<...>> will keep the context from persisting through all of the enclosing {...} sequence.

Any context can be kept alive by ensuring it has something to do at every musical moment. **Staff** contexts are kept alive by ensuring one of their voices is kept alive. One way of doing this is to add spacer rests to a voice in parallel with the real music. These need to be added to every **Voice** context which needs to be kept alive. If several voices are to be used sporadically it is safest to keep them all alive rather than attempting to rely on the exceptions mentioned above.

In the following example, both voice A and voice B are kept alive in this way for the duration of the piece:

```latex
\texttt{\textbf{musicA} = \relative { d''4 d d d }}
\texttt{musicB = \relative { g'4 g g g }}
\texttt{keepVoicesAlive = {}
\texttt{   << \new Voice = "A" \{ s1*5 \} \% Keep Voice "A" alive for 5 bars
\texttt{   \new Voice = "B" \{ s1*5 \} \% Keep Voice "B" alive for 5 bars
\texttt{   >>
\texttt{}}
\texttt{music = {}
\texttt{   \context Voice = "A" \{ \}
\texttt{   \voiceOneStyle
\texttt{   \musicA
\texttt{   \}
\texttt{   \context Voice = "B" \{ \}
\texttt{   \voiceTwoStyle
\texttt{   \musicB
\texttt{   \}
\texttt{   \context Voice = "A" \{ \musicA \}
\texttt{   \context Voice = "B" \{ \musicB \}
\texttt{   \context Voice = "A" \{ \musicA \}
\texttt{}}}
\texttt{\score {
\new Staff <<
\texttt{   \keepVoicesAlive
\texttt{   \music
\texttt{   >>
\texttt{}}
```

![Music notation example](image-url)
The following example shows how a sporadic melody line with lyrics might be written using this approach. In a real situation the melody and accompaniment would consist of several different sections, of course.

```latex
melody = \relative { a'4 a a a }
accompaniment = \relative { d'4 d d d }
words = \lyricmode { These words follow the melody }
\score {
  \context Voice = \midi Staff = "music" {
    \context Voice = \midi Staff = "melody" {
      \voiceOne 
      s1*4 % Keep Voice "melody" alive for 4 bars 
    }
    \context Voice = \midi Staff = "accompaniment" {
      \voiceTwo 
      \accompaniment 
    }
  }
  \context Voice = \midi Staff = "accompaniment" {
    \accompaniment 
  }
  \context Voice = \midi Staff = "melody" {
    \melody 
  }
  \context Voice = \midi Staff = "accompaniment" {
    \accompaniment 
  }
  \context Voice = \midi Staff = "melody" {
    \melody 
  }
  \context Voice = \midi Staff = "accompaniment" {
    \accompaniment 
  }
  \context Voice = \midi Staff = "melody" {
    \melody 
  }
  \context Voice = \midi Staff = "accompaniment" {
    \accompaniment 
  }
}
\new Lyrics \with { alignAboveContext = "music" }
\lyricsto "melody" { \words }
}
```

These words follow the melody

An alternative way, which may be better in many circumstances, is to keep the melody line alive by simply including spacer notes to line it up correctly with the accompaniment:

```latex
melody = \relative { 
  s1 % skip a bar 
  a'4 a a a 
  s1 % skip a bar 
  a4 a a a 
}
accompaniment = \relative { 
  d'4 d d d 
}
5.1.4 Modifying context plug-ins

Notation contexts (like Score and Staff) not only store properties, they also contain plug-ins called ‘engravers’ that create notation elements. For example, the Voice context contains a Note_heads_engraver and the Staff context contains a Key_engraver.

For a full description of each plug-in, see Internals Reference → Translation → Engravers. Every context described in Internals Reference → Translation → Context. lists the engravers used for that context.

It can be useful to shuffle around these plug-ins. This is done by starting a new context with \new or \context, and modifying it,

\new context \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}

\{ ...
  ...music...
}

where each ... should be the name of an engraver. Here is a simple example which removes Time_signature_engraver and Clef_engraver from a Staff context,
In the second staff there are no time signature or clef symbols. This is a rather crude method of making objects disappear since it will affect the entire staff. This method also influences the spacing, which may or may not be desirable. More sophisticated methods of blanking objects are shown in Section “Visibility and color of objects” in Learning Manual.

The next example shows a practical application. Bar lines and time signatures are normally synchronized across the score. This is done by the Timing_translator and Default_bar_line_engraver. This plug-in keeps an administration of time signature, location within the measure, etc. By moving these engraver from Score to Staff context, we can have a score where each staff has its own time signature.

\score {
<<
  \new Staff \relative { 
    f'2 g 
  }
  \new Staff \with { 
    \remove "Time_signature_engraver" 
    \remove "Clef_engraver" 
  } \relative { 
    f'2 g2 
  }
>>
}

\layout {
  \context { 
    \Score 
    \remove "Timing_translator" 
    \remove "Default_bar_line_engraver"
  }
}
Known issues and warnings

The order in which the engravers are specified is the order in which they are called to carry out their processing. Usually the order in which the engravers are specified does not matter, but in a few special cases the order is important, for example where one engraver writes a property and another reads it, or where one engraver creates a grob and another must process it.

The following orderings are important:

- the Bar_engraver must normally be first,
- the New_fingering_engraver must come before the Script_column_engraver,
- the Timing_translator must come before the Bar_number_engraver.

See also

Installed Files: ly/engraver-init.ly.

5.1.5 Changing context default settings

Context and grob properties can be changed with \set and \override commands, as described in Section 5.3 [Modifying properties], page 619. These commands create music events, making the changes take effect at the point in time the music is being processed.

In contrast, this section explains how to change the default values of context and grob properties at the time the context is created. There are two ways of doing this. One modifies the default values in all contexts of a particular type, the other modifies the default values in just one particular instance of a context.

Changing all contexts of the same type

The default context settings which are to be used for typesetting in Score, Staff, Voice and other contexts may be specified in a \context block within any \layout block.

Settings for Midi output as opposed to typesetting will have to be separately specified in \midi blocks (see [Output definitions - blueprints for contexts], page 597).

The \layout block should be placed within the \score block to which it is to apply, after the music.

\layout {
  \context {
    \Voice
    [context settings for all Voice contexts]
  }
  \context {
    \Staff
    [context settings for all Staff contexts]
The following types of settings may be specified:

- An \override command, but with the context name omitted

\score {
\relative {
  a'4^"Thicker stems" a a a
  a4 a a \ff a
}
\layout {
  \context {
    \Staff
    \override Stem.thickness = #4.0
  }
}
\timesig.C44/clefs.G

\text{Thicker stems}

- Directly setting a context property

\score {
\relative {
  a'4^"Smaller font" a a a
  a4 a a \ff a
}
\layout {
  \context {
    \Staff
    fontSize = #-4
  }
}
\timesig.C44/clefs.G

\text{Smaller font}

- A predefined command such as \dynamicUp or a music expression like \accidentalStyle
dodecaphonic

\score {
\relative {
  a'4^"Dynamics above" a a a
  a4 a a \ff a
}
\layout {
  \context {
    \Voice
    \dynamicUp
}
A user-defined variable containing a `\with` block; for details of the `\with` block see [Changing just one specific context], page 610.

```
StaffDefaults = \with {
  fontSize = #-4
}
```

Property-setting commands can be placed in a `\layout` block without being enclosed in a `\context` block. Such settings are equivalent to including the same property-setting commands at the start of every context of the type specified. If no context is specified every bottom-level context is affected, see [Bottom-level contexts - voices], page 599. The syntax of a property-setting command in a `\layout` block is the same as the same command written in the music stream.
Changing just one specific context

The context properties of just one specific context instance can be changed in a `\with` block. All other context instances of the same type retain the default settings built into LilyPond and modified by any `\layout` block within scope. The `\with` block must be placed immediately after the `\new context-type` command:

```
\new Staff \with { [context settings for this context instance only] }
{
  ...
}
```

Alternatively, if the music is being entered using the short form of the input mode-specifying commands, e.g. `\chords` rather than `\chordmode`, the `\with` command must be placed immediately after the mode-specifying command:

```
\chords \with { [context settings for this (implicit) context instance only] }
{
  ...
}
```

as it is the implicit context created by these short forms which should be modified. The same consideration applies to the other input mode-specifying short forms (`\drums, \figures`), see Section 5.4.1 [Input modes], page 632.

Since context modifications specified in `\with` blocks are inside music, they will affect all outputs (typesetting and Midi) as opposed to changes within an output definition.

The following types of settings may be specified:

- An `\override` command, but with the context name omitted

```latex
\score {
  \new Staff {
    \new Voice \with { \override Stem.thickness = #4.0 }
    {
      \relative {
        a'4^"Thick stems" a a a
        a4 a a a
      }
    }
  }
}
```

Thick stems
• Directly setting a context property

\score {
  \context {Staff {\with {\font {Default} fontsize = #-4}}}
  \new Staff {\new Voice {\relative {a'4"Default font" a a a a a a a}}}
  \new Staff {\with {\font {Smaller} fontsize = #-4}}
  \new Staff {\new Voice {\relative {a'4"Smaller font" a a a a a a a}}}
}

\context {Staff {\with {\font {Smaller} fontsize = #-4}}}
  \new Staff {\new Voice {\relative {a'4"Smaller font" a a a a a a a}}}
}

• A predefined command such as \dynamicUp

\score {
  \context {Staff {\with {\font {Default} fontsize = #-4}}}
  \new Staff {\new Voice {\relative {a'4"Dynamics below" a a a a a a \ff a}}}
  \new Staff {\with {\font {Smaller} fontsize = #-4}}
  \new Staff {\new Voice {\relative {a'4"Dynamics above" a a a a a a \ff a}}}
}

\context {Staff {\with {\font {Smaller} fontsize = #-4}}}
  \new Staff {\new Voice {\relative {a'4"Dynamics above" a a a a a a \ff a}}}
}

\context {Staff {\with {\font {Smaller} fontsize = #-4}}}
  \new Staff {\new Voice {\relative {a'4"Dynamics above" a a a a a a \ff a}}}
}
5.1.6 Defining new contexts

Specific contexts, like Staff and Voice, are made from simple building blocks. It is possible to create new types of contexts with different combinations of engraver plug-ins.

The next example shows how to build a different type of Voice context from scratch. It will be similar to Voice, but only prints centered slash note heads. It can be used to indicate improvisation in jazz pieces,

\begin{music}
\name ImproVoice
\context {
\layout {

ad lib
\!
undress
\!
\!
\!
\!
\!
\!
\!
\!
\!
\!
\!
while playing :)\}
\}
\end{music}

These settings are defined within a context block inside a layout block,

\begin{music}
\layout {
\context {

...\}
\}
\end{music}

In the following discussion, the example input shown should go in place of the ... in the previous fragment.

First it is necessary to define a name for the new context:

\begin{music}
\name ImproVoice
\end{music}

Since it is similar to the Voice context, we want commands that work in (existing) Voice contexts to continue working. This is achieved by giving the new context an alias of Voice,

\begin{music}
\alias Voice
\end{music}
The context will print notes and instructive texts, so we need to add the engravers which provide this functionality, plus the engraver which groups notes, stems and rests which occur at the same musical moment into columns,

```
\consists "Note_heads_engraver"
\consists "Text_engraver"
\consists "Rhythmic_column_engraver"
```

The note heads should all be placed on the center line,

```
\consists "Pitch_squash_engraver"
squashedPosition = #0
```

The `Pitch_squash_engraver` modifies note heads (created by the `Note_heads_engraver`) and sets their vertical position to the value of `squashedPosition`, in this case 0, the center line.

The notes look like a slash, and have no stem,

```
\override NoteHead.style = #'slash
\hide Stem
```

All these plug-ins have to communicate under the control of the context. The mechanisms with which contexts communicate are established by declaring the context `\type`. Within a `\layout` block, most contexts will be of type `Engraver_group`. Some special contexts and contexts in `\midi` blocks use other context types. Copying and modifying an existing context definition will also fill in the type. Since this example creates a definition from scratch, it needs to be specified explicitly.

```
\type "Engraver_group"
```

Put together, we get

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Pitch_squash_engraver"
  squashedPosition = #0
  \override NoteHead.style = #'slash
  \hide Stem
  \alias Voice
}
```

Contexts form hierarchies. We want to place the `ImproVoice` context within the `Staff` context, just like normal `Voice` contexts. Therefore, we modify the `Staff` definition with the `\accepts` command,

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Often when reusing an existing context definition, the resulting context can be used anywhere where the original context would have been useful.

```
\layout {
  ...
  \inherit-acceptability to from
}
```
will arrange to have contexts of type to accepted by all contexts also accepting from. For example, using

```
\layout {
  ...
  \inherit-acceptability "ImproVoice" "Voice"
}
```

will add an `\accepts` for `ImproVoice` to both `Staff` and `RhythmicStaff` definitions.

The opposite of `\accepts` is `\denies`, which is sometimes needed when reusing existing context definitions.

Arranging the required pieces into a `\layout` block leaves us with

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \inherit-acceptability "ImproVoice" "Voice"
}
```

Then the output at the start of this subsection can be entered as

```
\relative {
  a'4 d8 bes8
  \new ImproVoice {
    c4"ad lib" c
    c4 c"undress"
    c c_"while playing :)
  }
  a1
}
```

To complete this example, changes affecting the context hierarchy should be repeated in a `\midi` block so that Midi output depends on the same context relations.

**See also**


### 5.1.7 Context layout order

Contexts are normally positioned in a system from top to bottom in the order in which they are encountered in the input file. When contexts are nested, the outer context will include inner nested contexts as specified in the input file, provided the inner contexts are included in the outer context’s “accepts” list. Nested contexts which are not included in the outer context’s “accepts” list will be repositioned below the outer context rather than nested within it.

The “accepts” list of a context can be changed with the `\accepts` or `\denies` commands. `\accepts` adds a context to the “accepts” list and `\denies` removes a context from the list.

For example, a square-braced staff group is not usually found within a curved-braced staff with connecting staff bars, and a `GrandStaff` does not accept a `StaffGroup` inside it by default.

```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
}
```
However, by using the \accepts command, StaffGroup can be added to the GrandStaff context:

\score {
  \new GrandStaff <<
  \new StaffGroup <<
    \new Staff { c'1 } \
    \new Staff { d'1 } \\
    \new Staff { \set Staff.instrumentName = bottom f'1 } \\
  >>
  \new Staff { \set Staff.instrumentName = bottom f'1 } \\
  >>
  \layout {
    \context {
      \GrandStaff \\
      \accepts "StaffGroup"
    }
  }
}

\denies is mainly used when a new context is being based on another, but the required nesting differs. For example, the VaticanaStaff context is based on the Staff context, but with the VaticanaVoice context substituted for the Voice context in the “accepts” list.

Note that a context will be silently created implicitly if a command is encountered when there is no suitable context available to contain it.
Within a context definition, the type of subcontext to be implicitly created is specified using `\defaultchild`. A number of music events require a 'Bottom' context: when such an event is encountered, subcontexts are created recursively until reaching a context with no 'defaultchild' setting.

Implicit context creation can at times give rise to unexpected new staves or scores. Using `\new` to create contexts explicitly avoids those problems.

Sometimes a context is required to exist for just a brief period, a good example being the staff context for an ossia. This is usually achieved by introducing the context definition at the appropriate place in parallel with corresponding section of the main music. By default, the temporary context will be placed below all the existing contexts. To reposition it above the context called “main”, it should be defined like this:

```latex
\new Staff \with { alignAboveContext = "main" }
```

A similar situation arises when positioning a temporary lyrics context within a multi-staff layout such as a ChoirStaff, for example, when adding a second verse to a repeated section. By default the temporary lyrics context will be placed beneath the lower staves. By defining the temporary lyrics context with `alignBelowContext` it can be positioned correctly beneath the (named) lyrics context containing the first verse.

Examples showing this repositioning of temporary contexts can be found elsewhere — see Section “Nesting music expressions” in Learning Manual, Section 1.6.2 [Modifying single staves], page 199, and Section 2.1.2 [Techniques specific to lyrics], page 287.

**See also**


Notation Reference: Section 1.6.2 [Modifying single staves], page 199, Section 2.1.2 [Techniques specific to lyrics], page 287.

Application Usage: Section “An extra staff appears” in Application Usage.

Installed Files: 1y/engraver-init.ly.

### 5.2 Explaining the Internals Reference

#### 5.2.1 Navigating the program reference

Suppose we want to move the fingering indication in the fragment below:

```
c'' \times\times 2
\clef G
\noteheads s2
```

If you visit the documentation on fingering instructions (in [Fingering instructions], page 229), you will notice:

**See also**

Internals Reference: Section “Fingering” in Internals Reference.

The programmer’s reference is available as an HTML document. It is highly recommended that you read it in HTML form, either online or by downloading the HTML documentation. This section will be much more difficult to understand if you are using the PDF manual.

Follow the link to Section “Fingering” in Internals Reference. At the top of the page, you will see

Fingering objects are created by: Section “Fingering_engraver” in Internals Reference and Section “New_fingering_engraver” in Internals Reference.
By following related links inside the program reference, we can follow the flow of information within the program:

- Section “Fingering” in Internals Reference: Section “Fingering” in Internals Reference objects are created by: Section “Fingering_engraver” in Internals Reference
- Section “Fingering_engraver” in Internals Reference: Music types accepted: Section “fingering-event” in Internals Reference
- Section “fingering-event” in Internals Reference: Music event type fingering-event is in Music expressions named Section “FingeringEvent” in Internals Reference

This path goes against the flow of information in the program: it starts from the output, and ends at the input event. You could also start at an input event, and read with the flow of information, eventually ending up at the output object(s).

The program reference can also be browsed like a normal document. It contains chapters on Music definitions on Section “Translation” in Internals Reference, and the Section “Backend” in Internals Reference. Every chapter lists all the definitions used and all properties that may be tuned.

### 5.2.2 Layout interfaces

The HTML page that we found in the previous section describes the layout object called Section “Fingering” in Internals Reference. Such an object is a symbol within the score. It has properties that store numbers (like thicknesses and directions), but also pointers to related objects. A layout object is also called a Grob, which is short for Graphical Object. For more details about Grobs, see Section “grob-interface” in Internals Reference.

The page for Fingering lists the definitions for the Fingering object. For example, the page says

```
padding (dimension, in staff space):
0.5
```

which means that the number will be kept at a distance of at least 0.5 of the note head.

Each layout object may have several functions as a notational or typographical element. For example, the Fingering object has the following aspects

- Its size is independent of the horizontal spacing, unlike slurs or beams.
- It is a piece of text. Granted, it is usually a very short text.
- That piece of text is typeset with a font, unlike slurs or beams.
- Horizontally, the center of the symbol should be aligned to the center of the note head.
- Vertically, the symbol is placed next to the note and the staff.
- The vertical position is also coordinated with other superscript and subscript symbols.

Each of these aspects is captured in so-called interfaces, which are listed on the Section “Fingering” in Internals Reference page at the bottom.

This object supports the following interfaces: Section “item-interface” in Internals Reference, Section “self-alignment-interface” in Internals Reference, Section “side-position-interface” in Internals Reference, Section “text-interface” in Internals Reference, Section “text-script-interface” in Internals Reference, Section “font-interface” in Internals Reference, Section “finger-interface” in Internals Reference, and Section “grob-interface” in Internals Reference.

Clicking any of the links will take you to the page of the respective object interface. Each interface has a number of properties. Some of them are not user-serviceable (‘Internal properties’), but others can be modified.
We have been talking of the `Fingering` object, but actually it does not amount to much. The initialization file (see Section “Other sources of information” in Learning Manual) `scm/define-grobs.scm` shows the soul of the ‘object’,

```
(Fingering
 . ((padding 0.5)
   (avoid-slur around)
   (slur-padding 0.2)
   (staff-padding 0.5)
   (self-alignment-X 0)
   (self-alignment-Y 0)
   (script-priority 100)
   (stencil ,ly:text-interface::print)
   (direction ,ly:script-interface::calc-direction)
   (font-encoding fetaText)
   (font-size -5) ; don’t overlap when next to heads.
   (meta ((class Item)
           (interfaces (finger-interface
                         font-interface
                         text-script-interface
                         text-interface
                         side-position-interface
                         self-alignment-interface
                         item-interface))))))
```

As you can see, the `Fingering` object is nothing more than a bunch of variable settings, and the webpage in the Internals Reference is directly generated from this definition.

### 5.2.3 Determining the grob property

Recall that we wanted to change the position of the 2 in

\[ c''-2 \]

Since the 2 is vertically positioned next to its note, we have to meddle with the interface associated with this positioning. This is done using `side-position-interface`. The page for this interface says

```
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?)
```

Below this description, the variable `padding` is described as

```
padding (dimension, in staff space)
Add this much extra space between objects that are next to each other.
```

By increasing the value of `padding`, we can move the fingering away from the note head. The following command will insert “three staff spaces” worth of distance between the note and a fingering mark:

\[
\text{\textbackslash once \textbackslash override Voice.Fingering.padding = #3}
\]
Inserting the padding before the fingering object is created results in the following:

\once \override Voice.Fingering.padding = #3
c''-2

\[ \begin{array}{c}
\text{\begin{music}\ musica\end{music}}
\end{array} \]

In this case, the context for this tweak is \texttt{Voice}. See Section “Fingering\_engraver” in Internals Reference plug-in, which says:

Fingering\_engraver is part of contexts: ... Section “Voice” in Internals Reference

5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (also includes one-word names)
- LilyPond-specific scheme functions: ly:plus-scheme-style
- music events, music classes and music properties: as-scheme-functions
- Grob interfaces: scheme-style
- backend properties: scheme-style (but X and Y!)
- contexts (and MusicExpressions and grobs): Capitalized or CamelCase
- context properties: lowercaseFollowedByCamelCase
- engravers: Capitalized\_followed\_by\_lowercase\_and\_with\_underscores

Questions to be answered:

- Which of these are conventions and which are rules?
- Which are rules of the underlying language, and which are LilyPond-specific?

5.3 Modifying properties

5.3.1 Overview of modifying properties

Each context is responsible for creating certain types of graphical objects. The settings used for printing these objects are also stored by context. By changing these settings, the appearance of objects can be altered.

There are two different kinds of properties stored in contexts: context properties and grob properties. Context properties are properties that apply to the context as a whole and control how the context itself is displayed. In contrast, grob properties apply to specific grob types that will be displayed in the context.

The \texttt{\set} and \texttt{\unset} commands are used to change values for context properties. The \texttt{\override} and \texttt{\revert} commands are used to change values for grob properties.

See also

Internals Reference: Section “Backend” in Internals Reference, Section “All layout objects” in Internals Reference, Section “OverrideProperty” in Internals Reference, Section “RevertProperty” in Internals Reference, Section “PropertySet” in Internals Reference.

Known issues and warnings

The back-end is not very strict in type-checking object properties. Cyclic references in Scheme values for properties can cause hangs or crashes, or both.
5.3.2 The `\set` command

Each context has a set of properties, variables contained in that context. Context properties are changed with the `\set` command, which has the following syntax:

```
\set context.property = #value
```

`value` is a Scheme object, which is why it must be preceded by the `#` character.

Contexts properties are usually named in `studlyCaps`. They mostly control the translation from music to notation, e.g., `localAlterations` (for determining whether to print accidentals), or `measurePosition` (for determining when to print a bar line). Context properties can change value over time while interpreting a piece of music; `measurePosition` is an obvious example of this. Context properties are modified with `\set`.

For example, multimeasure rests will be combined into a single bar (as explained in [Compressing empty measures], page 222) if the context property `skipBars` is set to `#t`:

```
R1*2
\set Score.skipBars = ##t
R1*2
```

If the `context` argument is left out, then the property will be set in the current bottom context (typically `ChordNames`, `Voice`, `TabVoice`, or `Lyrics`).

```
\set Score.autoBeaming = ##f
\relative {
  e''8 e e e
  \set autoBeaming = ##t
  e8 e e e
}
\relative {
  c''8 c c c c8 c c c
}
```

The change is applied ‘on-the-fly’, during the music, so that the setting only affects the second group of eighth notes.

Note that the bottom-most context does not always contain the property that you wish to change – for example, attempting to set the `skipBars` property of the default bottom context, in this case `Voice`, will have no effect, because `skipBars` is a property of the `Score` context.

```
R1*2
\set skipBars = ##t
R1*2
```

Contexts are hierarchical, so if an enclosing context was specified, for example `Staff`, then the change would also apply to all `Voices` in the current staff.
The \unset command:

\unset context.property

is used to remove the definition of property from context. This command removes the definition only if it is set in context. Properties that have been set in enclosing contexts will not be altered by an unset in an enclosed context:

\set Score.autoBeaming = ##t
relative {
  \unset autoBeaming
e''8 e e e
  \unset Score.autoBeaming
e8 e e e
}
\relative {
  c''8 c c c c8 c c c
}

Like \set, the context argument does not have to be specified for a bottom context, so the two statements

\set Voice.autoBeaming = ##t
\set autoBeaming = ##t

are equivalent if the current bottom context is Voice.

Preceding a \set or \unset command by \once makes the setting apply to only a single time-step:

c''4
\once \set fontSize = #4.7
c''4
c''4

A full description of all available context properties is in the internals reference, see Translation \mapsto Tunable context properties.

See also

Internals Reference: Section “Tunable context properties” in Internals Reference.

5.3.3 The \override command

There is a special type of context property: the grob description. Grob descriptions are named in StudlyCaps (starting with capital letters). They contain the ‘default settings’ for a particular kind of grob as an association list. See scm/define-grobs.scm to see the settings for each grob description. Grob descriptions are modified with \override.

The syntax for the \override command is

\override [context.]GrobName.property = #value
For example, we can increase the thickness of a note stem by overriding the `thickness` property of the `Stem` object:

\[\texttt{\textbackslash override Voice.Stem.thickness = \#3.0}\]

\[
\begin{array}{c}
\texttt{c''4 c''}
\end{array}
\]

If no context is specified in an `\override`, the bottom context is used:

\[\texttt{\textbackslash override Staff.Stem.thickness = \#3.0}\]

\[
\begin{array}{c}
\texttt{\relative { e''4 e e4 e}}
\end{array}
\]

Some tweakable options are called ‘subproperties’ and reside inside properties. To tweak those, use commands in the form

\[\texttt{\textbackslash override Stem.details.beamed-lengths = \#'(4 4 3)}\]

or to modify the ends of spanners, use a form like these

\[\texttt{\textbackslash override TextSpanner.bound-details.left.text = "left text" \textbackslash override TextSpanner.bound-details.right.text = "right text"}\]

The effects of `\override` can be undone by `\revert`.

The syntax for the `\revert` command is

\[\texttt{\revert [context.]GrobName.property}\]

For example,

\[
\begin{array}{c}
\texttt{\relative { \texttt{c''4 c''}}}
\end{array}
\]

\[
\begin{array}{c}
\texttt{\textbackslash override Voice.Stem.thickness = \#3.0}
\end{array}
\]

\[
\begin{array}{c}
\texttt{\textbackslash revert Voice.Stem.thickness}
\end{array}
\]

\[
\begin{array}{c}
\texttt{c4 c}
\end{array}
\]

\[
\begin{array}{c}
\texttt{\relative { c'4 c c c}}
\end{array}
\]

\[
\begin{array}{c}
\texttt{\textbackslash revert Voice.Stem.thickness}
\end{array}
\]

\[
\begin{array}{c}
\texttt{\textbackslash revert Voice.Stem.thickness}
\end{array}
\]
The effects of \override and \revert apply to all grobs in the affected context from the current time forward:

\[ \begin{align*}
&\relative {
  e^\textprime'4 \\
  \override Staff.Stem.thickness = #3.0 \\
  e4 e e \\
}\}

&\relative {
  c^\textprime'4 c c \\
  \revert Staff.Stem.thickness \\
  c4 \\
}\}
\end{align*} \]

\once can be used with \override or \revert to affect only the current time step:

\[ \begin{align*}
&\relative c {
  e^\textprime'4 e e e \\
}\}

&\relative {
  c^\textprime'4 \\
  \once \override Staff.Stem.thickness = #3.0 \\
  c4 c c \\
}\}
\end{align*} \]

See also

Internals Reference: Section “Backend” in Internals Reference

5.3.4 The \tweak command

Changing grob properties with \override causes the changes to apply to all of the given grobs in the context at the moment the change applies. Sometimes, however, it is desirable to have changes apply to just one grob, rather than to all grobs in the affected context. This is accomplished with the \tweak command, which has the following syntax:

\tweak [layout-object.]grob-property value

Specifying layout-object is optional. The \tweak command applies to the music object that immediately follows value in the music stream.

For an introduction to the syntax and uses of the tweak command see Section “Tweaking methods” in Learning Manual.
When several similar items are placed at the same musical moment, the \override command cannot be used to modify just one of them – this is where the \tweak command must be used. Items which may appear more than once at the same musical moment include the following:

- note heads of notes inside a chord
- articulation signs on a single note
- ties between notes in a chord
- tuplet brackets starting at the same time

In this example, the color of one note head and the type of another note head are modified within a single chord:

\begin{verbatim}
< c''
\tweak color #red
d''
g''
\tweak duration-log #1
a''
> 4
\end{verbatim}

\tweak can be used to modify slurs:

\begin{verbatim}
relative { c'-\tweak thickness #5 ( d e f ) }
\end{verbatim}

For the \tweak command to work, it must remain immediately adjacent to the object to which it is to apply after the input file has been converted to a music stream. Tweaking a whole chord does not do anything since its music event only acts as a container, and all layout objects are created from events inside of the EventChord:

\begin{verbatim}
\tweak color #red c''4
\tweak color #red <c'' e''>4
<\tweak color #red c'' e''>4
\end{verbatim}

The simple \tweak command cannot be used to modify any object that is not directly created from the input. In particular it will not affect stems, automatic beams or accidentals, since these are generated later by NoteHead layout objects rather than by music elements in the input stream.

Such indirectly created layout objects can be tweaked using the form of the \tweak command in which the grob name is specified explicitly:

\begin{verbatim}
\tweak Stem.color #red
\tweak Beam.color #green c''8 e''
<c'' e'' \tweak Accidental.font-size #-3 ges'>4
\end{verbatim}
\tweak cannot be used to modify clefs or time signatures, since these become separated from any preceding \tweak command in the input stream by the automatic insertion of extra elements required to specify the context.

Several \tweak commands may be placed before a notational element – all affect it:

\c'\-
\tweak style #'dashed-line
\tweak dash-fraction #0.2
\tweak thickness #3
\tweak color #red
\glissando
\f''

The music stream which is generated from a section of an input file, including any automatically inserted elements, may be examined, see Section “Displaying music expressions” in Extending. This may be helpful in determining what may be modified by a \tweak command, or in determining how to adjust the input to make a \tweak apply.

See also

Extending LilyPond: Section “Displaying music expressions” in Extending.

Known issues and warnings

The \tweak command cannot be used to modify the control points of just one of several ties in a chord, other than the first one encountered in the input file.

5.3.5 \set vs. \override

The \set and \override commands manipulate properties associated with contexts. In both cases, the properties follow a hierarchy of contexts: properties that are not set themselves in a context will still show the values of their respective parent’s context.

The lifetime and value of a context property is dynamic and only available when music is being interpreted (i.e., ‘iterated’). At the time of the context’s creation, properties are initialized from its corresponding definitions (along with any other modifications) of that context. Any subsequent changes are achieved with any ‘property-setting’ commands that are within the music itself.

Graphical Object (or “grob”) definitions are a special category of context properties as their structure and use is different from that of normal context properties. Unlike normal context properties, grob definitions are subdivided into grob properties.

Also, in contrast to normal context properties, grob definitions have their own internal ‘bookkeeping’ used to keep track of their own individual grob properties and any sub-properties. This means that it is possible to define those parts within different contexts and yet still have the overall grob definition at the time of grob creation from all the pieces provided amongst the current context and its parent(s).

A grob is usually created by an engraver at the time of interpreting a music expression and receives its initial properties from the current grob definition of the engraver’s context.
engraver (or other ‘backend’ parts of LilyPond) can then change (or add to) the grob’s initial properties. However, this does not affect the context’s own grob definition.

What LilyPond calls grob properties in the context of ‘user-level’ tweaks are really the properties of a context’s own grob definition.

Grob definitions are accessed with a different set of commands and are manipulated using \override and \revert and have a name starting with a capital letter (e.g., ‘NoteHead’); whereas normal context properties are manipulated using \set and \unset and are named starting with a lowercase letter.

The commands \tweak and \overrideProperty change grob properties by bypassing all context properties completely and, instead, catch grobs as they are being created, setting properties on them for a music event (\tweak) or, in the case of \overrideProperty for a specific override.

5.3.6 The \offset command

While it is possible to set grob properties to new values with the \override, \tweak, and \overrideProperty commands, it is often more convenient to modify such properties relative to a default value. The \offset command is available for this purpose.

The syntax for \offset is

\[-\]offset property offsets item

The command works by adding the contents of offsets to the default setting of the property of the grob indicated by item.

Depending on the formulation of the command, \offset may act as either a \tweak or \override. The variations in usage are discussed after consideration is given to grob properties that may be used with \offset.

Properties which may be offset

Many, but not all, grob properties may be offset. If property cannot be offset, the object will remain unchanged and a warning will be issued. In such cases, \override or \tweak should be used to modify the object instead.

One can work by trial and error and let the warnings be the guide to what may or may not be offset. A more systematic approach is possible, however.

The following criteria determine whether a property can be modified with \offset:

• The property has a ‘default setting’ in the grob’s description. Such properties are listed for each grob in Section “All layout objects” in Internals Reference. (They are also found in scm/define-grobs.scm.)

• The property takes a numerical value. Numerical values include number, list of numbers, number-pair, and number-pair-list. The pages at Section “All layout objects” in Internals Reference list the type of data characteristic to each property. It is immaterial whether the default setting is a function.

• The property cannot be a ‘subproperty’—a property residing within another property.

• Properties set to infinite values cannot be offset. There is no sensible way to offset positive and negative infinity.

The following examples consider several grob properties against the criteria outlined above.

• Properties that may be offset

Hairpin.height

This property is not a subproperty, and it is listed at Section “Hairpin” in Internals Reference. For a value, it takes ‘dimension, in staff space’ set to 0.6666—clearly a non-infinite number.
The page Section “Arpeggio” in Internals Reference lists a positions property which accepts a ‘pair of numbers’. It defaults to ly:arpeggio::positions—a callback which will be evaluated during the typesetting phase to yield a pair of numbers for any given Arpeggio object.

- Properties that may not be offset

**Hairpin.color**

There is no listing for color at Section “Hairpin” in Internals Reference.

**Hairpin.circled-tip**

The listing for Hairpin.circled-tip at Section “Hairpin” in Internals Reference shows that it takes a boolean value. Booleans are non-numerical.

**Stem.details.lengths**

Though listed at Section “Stem” in Internals Reference and defaulting to a list of numbers, this is a ‘subproperty’. There is currently no support for ‘nested properties’.

### offset as an override

If item is a grob name like Arpeggio or Staff.OttavaBracket, the result is an \override of the specified grob-type.

```latex
\offset property offsets [context.]GroName
```

Note that the leading hyphen is never used with the ‘override’ form, just as it is never used with the \override command itself.

The following example uses the ‘override’ form to lengthen the default arpeggios shown in the first measure to cover the extent of the chords more fully. The arpeggios are stretched by a half staff-space to top and bottom. Also shown is the same operation done on the first chord with an ordinary override of the positions property. This method is not at all expressive of the task of ‘stretching by a half staff-space’, as the endpoints must be specified with absolute rather than relative coordinates. Furthermore, individual overrides would be needed for the other chords, as they vary in size and position.

```latex
arpeggioMusic = {
    <c' e' g'>\arpeggio <a' c'' e''>\arpeggio
    <d' f' a' c''>\arpeggio <c' e' g' b' d'' f'' a''>\arpeggio
} {
    \arpeggioMusic
    \bar "||"
    \offset positions #'(-0.5 . 0.5) Arpeggio
    \arpeggioMusic
    \bar "||"
    \once \override Arpeggio.positions = #'(-3.5 . -0.5)
    <c' e' g'>\arpeggio
    \bar "||"
}
```

![Example music notation]
In its ‘override’ usage, \offset may be prefaced with \once or \temporary and reverted using \revert with property (see Section “Intermediate substitution functions” in Extending). This follows from the fact that \offset actually creates an \override of property:

\begin{verbatim}
music = { c'8\< d' e' f'\} 
{
  \music \offset height 1 Hairpin \music \music \revert Hairpin.height \music \bar "||"
  \once \offset height 1 Hairpin \music \music \bar "||"
  \override Hairpin.height = 0.2 \music \temporary \offset height 2 Hairpin \music \music \revert Hairpin.height \music \bar "||"
}
\end{verbatim}

Also like \override, the ‘override’ form of \offset may be used with \undo and \single.

\begin{verbatim}
longStem = \offset length 6 Stem 
{
  \longStem c'4 c''' c' c'' \bar "||"
  \undo \longStem c'4 c''' c' c'' \bar "||"
  \single \longStem c'4 c''' c' c'' \bar "||"
}
\end{verbatim}
\offset as a tweak

If item is a music expression such as ( or arpeggio, the result is the same music expression with a tweak applied.

[-]\offset [GrobName.]property offsets music-expression

The syntax of \offset in its ‘tweak’ form is analogous to the \tweak command itself, both in ordering and in the presence or absence of the leading hyphen.

The following example uses the ‘tweak’ form to adjust the vertical position of the BreathingSign object. Compare this with the ordinary \tweak command also demonstrated. The syntax is equivalent; however, the output of \tweak is less intuitive, since BreathingSign.Y-offset is calculated from the middle staff-line. It is not necessary to know how Y-offset is calculated when using \offset.

{\[c''4}\breathe\]c''4\offset Y-offset 2 \breathe\]c''2\tweak Y-offset 3 \breathe\}

In the previous example, the tweaked objects were created directly from the user input: the \breathe command was an explicit instruction to return a BreathingSign object. Since the focus of the command was unambiguous, there was no need to specify the object’s name. When an object is indirectly created, however, it is necessary to include the grob’s name. This is the same as for the \tweak command.

In the following example, the Beam object is lowered two staff-spaces by applying \offset to the positions property.

The first application of \offset requires that the grob’s name be included, because nothing in the input explicitly creates the beam. In the second application, the beam is created manually with the music expression [; therefore, the grob’s name is not needed. (Also illustrated is a shorthand: a single number will be applied to both members of a number-pair.)

{\[c''8 g'' e'' d''\offset Beam.positions #'(-2 . -2)\]c''8 g'' e'' d''c''8 g'' e'' d''c''8-\offset positions #-2 [ g'' e'' d'']\}

In the previous example, the tweaked objects were created directly from the user input: the \breathe command was an explicit instruction to return a BreathingSign object. Since the focus of the command was unambiguous, there was no need to specify the object’s name. When an object is indirectly created, however, it is necessary to include the grob’s name. This is the same as for the \tweak command.

In the following example, the Beam object is lowered two staff-spaces by applying \offset to the positions property.

The first application of \offset requires that the grob’s name be included, because nothing in the input explicitly creates the beam. In the second application, the beam is created manually with the music expression [; therefore, the grob’s name is not needed. (Also illustrated is a shorthand: a single number will be applied to both members of a number-pair.)
\textbf{offset with broken spanners}

Independently modifying segments of a spanner extending over a line break or breaks is also possible. In this case, \texttt{offsets} takes a list of values of the property’s required data type.

The \texttt{offset} command used in this manner is similar to the \texttt{alterBroken} command. (See Section 5.5.5 [Modifying broken spanners], page 659.) In contrast with \texttt{alterBroken}, however, the values given to \texttt{offset} are relative, not absolute.

The following example displaces the ‘broken’ \texttt{OttavaBracket} object through its \texttt{staff-padding} property. Since the property takes a \texttt{number}, \texttt{offsets} is provided with a list of \texttt{numbers} to account for the two segments created by the line break. The bracket piece on the first line is effectively untouched since 0 is added to its default value of \texttt{staff-padding}. The segment on the second line is raised three staff-spaces from its default height. The default height happens to be 2, though it is not necessary to know this to achieve the desired positioning.

\begin{verbatim}
{ 
  \offset staff-padding #'(0 3) Staff.OttavaBracket
  \ottava #1 
  c''''2 c''' 
  \break 
  c''''2 c''' 
}
\end{verbatim}

The following example mimics the effect of the \texttt{shape} command by offsetting the \texttt{control-points} property of the \texttt{Slur} object. Here, \texttt{offsets} is a list of \texttt{number-pair-lists}, one for each slur segment. This example achieves a result identical to the corresponding illustration at Section 5.5.4 [Modifying shapes], page 655.

\begin{verbatim}
{ 
  c'4-\offset control-points #'( 
    ((0 . 0) (0 . 0) (0 . 0) (0 . 1)) 
    ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
  ) ( f'4 g' c'' 
  \break 
  d'4 c'' f' c')
}
\end{verbatim}
5.3.7 Modifying alists

Some user-configurable properties are internally represented as alists (association lists), which store pairs of keys and values. The structure of an alist is:

```
'((key1 . value1)
 (key2 . value2)
 (key3 . value3)
 ...)
```

If an alist is a grob property or \texttt{paper} variable, its keys can be modified individually without affecting other keys.

For example, to reduce the space between adjacent staves in a staff-group, use the \texttt{staff-staff-spacing} property of the \texttt{StaffGrouper} grob. The property is an alist with four keys: \texttt{basic-distance}, \texttt{minimum-distance}, \texttt{padding}, and \texttt{stretchability}. The standard settings for this property are listed in the “Backend” section of the Internals Reference (see Section “StaffGrouper” in \texttt{Internals Reference}):

```
'((basic-distance . 9)
 (minimum-distance . 7)
 (padding . 1)
 (stretchability . 5))
```

One way to bring the staves closer together is by reducing the value of the \texttt{basic-distance} key (9) to match the value of \texttt{minimum-distance} (7). To modify a single key individually, use a \texttt{nested declaration}:

\begin{verbatim}
% default space between staves
\new PianoStaff <<
 \new Staff { \clef treble c''1 }
 \new Staff { \clef bass c1 }
>>

% reduced space between staves
\new PianoStaff \with {
 % this is the nested declaration
 \override StaffGrouper.staff-staff-spacing.basic-distance = #7
} <<
 \new Staff { \clef treble c''1 }
 \new Staff { \clef bass c1 }
>>
\end{verbatim}

Using a nested declaration will update the specified key (such as \texttt{basic-distance} in the above example) without altering any other keys already set for the same property.
Now suppose we want the staves to be as close as possible without overlapping. The simplest way to do this is to set all four alist keys to zero. However, it is not necessary to enter four nested declarations, one for each key. Instead, the property can be completely re-defined with one declaration, as an alist:

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
    (minimum-distance . 0)
    (padding . 0)
    (stretchability . 0))
} <<
\new Staff { \clef treble c''1 }
\new Staff { \clef bass c1 }
>>
```

Note that any keys not explicitly listed in the alist definition will be reset to their default-when-unset values. In the case of `staff-staff-spacing`, any unset key-values would be reset to zero (except `stretchability`, which takes the value of `basic-distance` when unset). Thus the following two declarations are equivalent:

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
    (minimum-distance . 0)
    (padding . 0)
    (stretchability . 7))
```

One (possibly unintended) consequence of this is the removal of any standard settings that are set in an initialization file and loaded each time an input file is compiled. In the above example, the standard settings for `padding` and `minimum-distance` (defined in `scm/define-grobs.scm`) are reset to their default-when-unset values (zero for both keys). Defining a property or variable as an alist (of any size) will always reset all unset key-values to their default-when-unset values. Unless this is the intended result, it is safer to update key-values individually with a nested declaration.

**Note:** Nested declarations will not work for context property alists (such as `beamExceptions`, `keyAlterations`, `timeSignatureSettings`, etc.). These properties can only be modified by completely re-defining them as alists.

### 5.4 Useful concepts and properties

#### 5.4.1 Input modes

The way in which the notation contained within an input file is interpreted is determined by the current input mode. In general, there are two ways of specifying the mode: a long form, e.g.
\chordmode, and a short form, e.g. \chords. The long form is typically used when supplying input to a variable or when entering input directly into an explicitly created context. The short form implicitly creates a context of the correct type for the input and passes the input directly to it. It is useful in simple situations when there is no requirement to explicitly create the receiving context.

**Chord mode**

This is activated with the \chordmode command, and causes input to be interpreted with the syntax of chord notation, see Section 2.7 [Chord notation], page 423. Music in chord mode is rendered as chords on a staff when entered into a Staff context, as chord names when entered into a ChordNames context or as fretboards when entered into a FretBoards context.

Chord mode is also activated with the \chords command. This also causes the following input to be interpreted with the syntax of chord notation but in addition it implicitly creates a new ChordNames context and renders the input into it as chord names, see [Printing chord names], page 429.

**Drum mode**

This is activated with the \drummode command, and causes input to be interpreted with the syntax of drum notation, see [Basic percussion notation], page 403. Music in drum mode is rendered as percussion notes when entered into a DrumStaff context.

Drum mode is also activated with the \drums command. This also causes the following input to be interpreted with the syntax of drum notation but in addition it implicitly creates a new DrumStaff context and renders the input into it as percussion notes, see [Basic percussion notation], page 403.

**Figure mode**

This is activated with the \figuremode command, and causes input to be interpreted with the syntax of figured bass, see [Entering figured bass], page 438. Music in figure mode is rendered as figured bass when entered into a FiguredBass context or a Staff context.

Figure mode is also activated with the \figures command. This also causes the following input to be interpreted with the figured bass syntax but in addition it implicitly creates a new FiguredBass context and renders the input into it as figured bass, see [Introduction to figured bass], page 437.

**Fret and tab modes**

There are no special input modes for entering fret and tab symbols.

To create tab diagrams, enter notes or chords in note mode and render them in a TabStaff context, see [Default tablatures], page 355.

To create fret diagrams above a staff, enter notes or chords in either note mode or chord mode and render them in a FretBoards context, see [Automatic fret diagrams], page 393. Alternatively, fret diagrams can be entered as markup above the notes using the \fret-diagram command, see [Fret diagram markups], page 373.

**Lyrics mode**

This is activated with the \lyricmode command, and causes input to be interpreted as lyric syllables with optional durations and associated lyric modifiers, see Section 2.1 [Vocal music], page 274. Input in lyric mode is rendered as lyric syllables when entered into a Lyrics context.

Lyric mode is also activated with the \lyrics command. This also causes the following input to be interpreted as lyric syllables but in addition it implicitly creates a new Lyrics context and renders the input into it as lyric syllables.
Lyric mode is also activated with the \addlyrics command. This also implicitly creates a new Lyrics context and in addition it adds an implicit \lyricsto command which associates the following lyrics with the preceding music, see [Automatic syllable durations], page 278.

**Markup mode**

This is activated with the \markup command, and causes input to be interpreted with the syntax of markup, see Section A.11 [Text markup commands], page 711.

**Note mode**

This is the default mode or it may be activated with the \notemode command. Input is interpreted as pitches, durations, markup, etc and typeset as musical notation on a staff.

It is not normally necessary to specify note mode explicitly, but it may be useful to do so in certain situations, for example if you are in lyric mode, chord mode or any other mode and want to insert something that only can be done with note mode syntax.

### 5.4.2 Direction and placement

In typesetting music the direction and placement of many items is a matter of choice. For example, the stems of notes can be directed up or down; lyrics, dynamics, and other expressive marks may be placed above or below the staff; text may be aligned left, right or center; etc. Most of these choices may be left to be determined automatically by LilyPond, but in some cases it may be desirable to force a particular direction or placement.

**Articulation direction indicators**

By default some directions are always up or always down (e.g., dynamics or fermata), while other things can alternate between up or down based on the stem direction (like slurs or accents).

The default action may be overridden by prefixing the articulation by a *direction indicator*. Three direction indicators are available: `^` (meaning “up”), `_` (meaning “down”) and `-` (meaning “use default direction”). The direction indicator can usually be omitted, in which case `-` is assumed, but a direction indicator is **always** required before:

- `\tweak` commands
- `\markup` commands
- `\tag` commands
- string markups, e.g., , "string"
- fingering instructions, e.g., `-1`
- articulation shortcuts, e.g., `, , ->, --`

Direction indicators affect only the next note:

```latex
\relative {
    c' '2( c)
    c2_.( c)
    c2( c)
    c2^*( c)
}
```

![Musical notation example]
The direction property
The position or direction of many layout objects is controlled by the `direction` property.

The value of the `direction` property may be set to 1, meaning “up” or “above”, or to -1, meaning “down” or “below”. The symbols UP and DOWN may be used instead of 1 and -1 respectively. The default direction may be specified by setting `direction` to 0 or CENTER. Alternatively, in many cases predefined commands exist to specify the direction. These are of the form

```
\xxxUp, \xxxDown or \xxxNeutral
```

where `\xxxNeutral` means “use the default” direction. See Section “Within-staff objects” in Learning Manual.

In a few cases, arpeggio for example, the value of the `direction` property can specify whether the object is to be placed to the right or left of the parent. In this case -1 or LEFT means “to the left” and 1 or RIGHT means “to the right”. 0 or CENTER means “use the default” direction.

These indications affect all notes until they are canceled.

```
\relative {
  c'2( c)
  \slurDown
  c2( c)
  c2( c)
  \slurNeutral
  c2( c)
}
```

In polyphonic music, it is generally better to specify an explicit voice than change an object’s direction. For more information, see Section 1.5.2 [Multiple voices], page 173.

See also


Notation Reference: Section 1.5.2 [Multiple voices], page 173.

5.4.3 Distances and measurements
Distances in LilyPond are of two types: absolute and scaled.

Absolute distances are used for specifying margins, indents, and other page layout details, and are by default specified in millimeters. Distances may be specified in other units by following the quantity by `\mm, \cm, \in` (inches), or `\pt` (points, 1/72.27 of an inch). Page layout distances can also be specified in scalable units (see the following paragraph) by appending `\staff-space` to the quantity. Page layout is described in detail in Section 4.1 [Page layout], page 546.

Scaled distances are always specified in units of the staff-space or, rarely, the half staff-space. The staff-space is the distance between two adjacent staff lines. The default value can be changed globally by setting the global staff size, or it can be overridden locally by changing the `staff-space` property of StaffSymbol. Scaled distances automatically scale with any change to the either the global staff size or the `staff-space` property of StaffSymbol, but fonts scale automatically only with changes to the global staff size. The global staff size thus enables the overall size of a rendered score to be easily varied. For the methods of setting the global staff size see Section 4.2.2 [Setting the staff size], page 559.
If just a section of a score needs to be rendered to a different scale, for example an ossia section or a footnote, the global staff size cannot simply be changed as this would affect the entire score. In such cases the change in size is made by overriding both the \staff-space property of \StaffSymbol and the size of the fonts. A Scheme function, \magstep, is available to convert from a font size change to the equivalent change in \staff-space. For an explanation and an example of its use, see Section “Length and thickness of objects” in Learning Manual.

**See also**


Notation Reference: Section 4.1 [Page layout], page 546, Section 4.2.2 [Setting the staff size], page 559.

### 5.4.4 Dimensions

The dimensions of a graphical object specify the positions of the left and right edges and the bottom and top edges of the objects’ bounding box as distances from the objects’ reference point in units of staff-spaces. These positions are usually coded as two Scheme pairs. For example, the text markup command \with-dimensions takes three arguments, the first two of which are a Scheme pair giving the left and right edge positions and a Scheme pair giving the bottom and top edge positions:

\[
\text{\with-dimensions} \ '(-5 \ 10) \ '(-3 \ 15) \ \text{arg}
\]

This specifies a bounding box for \text{arg} with its left edge at -5, its right edge at 10, its bottom edge at -3 and its top edge at 15, all measured from the objects’ reference point in units of staff-spaces.

**See also**

Notation Reference: Section 5.4.3 [Distances and measurements], page 635.

### 5.4.5 Staff symbol properties

The vertical position of staff lines and the number of staff lines can be defined at the same time. As the following example shows, note positions are not influenced by the staff line positions.

Note: The \line-positions property overrides the \line-count property. The number of staff lines is implicitly defined by the number of elements in the list of values for \line-positions.

\[
\text{\new Staff} \ \text{\with} \{
\text{\override StaffSymbol.line-positions} = \ '#'(7 3 0 -4 -6 -7)
\}
\text{\relative}{a4 e' f b | d1}
\]

The width of a staff can be modified. The units are staff spaces. The spacing of objects inside the staff is not affected by this setting.

\[
\text{\new Staff} \ \text{\with} \{
\text{\override StaffSymbol.width} = \#23
\}
\text{\relative}{a4 e' f b | d1}
\]
5.4.6 Spanners

Many objects of musical notation extend over several notes or even several bars. Examples are slurs, beams, tuplet brackets, volta repeat brackets, crescendi, trills, and glissandi. Such objects are collectively called “spanners”, and have special properties to control their appearance and behaviour. Some of these properties are common to all spanners; others are restricted to a sub-set of the spanners.

All spanners support the **spanner-interface**. A few, essentially those that draw a straight line between the two objects, support in addition the **line-spanner-interface**.

**Using the spanner-interface**

This interface provides two properties that apply to several spanners.

**The minimum-length property**

The minimum length of the spanner is specified by the **minimum-length** property. Increasing this usually has the necessary effect of increasing the spacing of the notes between the two end points. However, this override has no effect on many spanners, as their length is determined by other considerations. A few examples where it is effective are shown below.

\begin{verbatim}
\relative \compressMMRests {
  a'\tie a'
  a'
  \tweak minimum-length #5
  a'
\}
\end{verbatim}

\begin{verbatim}
\relative \compressMMRests {
  a' \tie a a a \!
  \tweak minimum-length #20
  a \tie a a a \!
\}
\end{verbatim}
This override can also be used to increase the length of slurs and phrasing slurs:
\relative {
  a'( g)
a\tweak minimum-length \#5
  ( g)
a\( g\)
a\tweak minimum-length \#5
  \( g\)
}

For some layout objects, the minimum-length property becomes effective only if the set-spacing-rods procedure is called explicitly. To do this, the springs-and-rods property should be set to ly:spanner::set-spacing-rods. For example, the minimum length of a glissando has no effect unless the springs-and-rods property is set:

\%
% default
\once \override Glissando.minimum-length = \#20
\override Beam.minimum-length = \#20
\%
% not effective alone
\override Glissando.minimum-length = \#20
\override Beam.minimum-length = \#20
\%
% effective only when both overrides are present
\override Glissando.springs-and-rods = ly:spanner::set-spacing-rods
\override Beam.springs-and-rods = ly:spanner::set-spacing-rods

The same is true of the Beam object:

\%
% not effective alone
\once \override Beam.minimum-length = \#20
\%
% effective only when both overrides are present
\once \override Beam.springs-and-rods = ly:spanner::set-spacing-rods
**The to-barline property**

The second useful property of the spanner-interface is to-barline. By default this is true, causing hairpins and other spanners which are terminated on the first note of a measure to end instead on the immediately preceding bar line. If set to false, the spanner will extend beyond the bar line and end on the note itself:

```latex
\relative { 
a' \! a a a \! a a a 
\override Hairpin.to-barline = ##f
a \! a a a \! a a a
}
```

This property is not effective for all spanners. For example, setting it to `##t` has no effect on slurs or phrasing slurs or on other spanners for which terminating on the bar line would not be meaningful.

**Using the line-spanner-interface**

Objects which support the line-spanner-interface include
- DynamicTextSpanner
- Glissando
- TextSpanner
- TrillSpanner
- VoiceFollower

The routine responsible for drawing the stencils for these spanners is `ly:line-spanner::print`. This routine determines the exact location of the two end points and draws a line between them, in the style requested. The locations of the two end points of the spanner are computed on-the-fly, but it is possible to override their Y-coordinates. The properties which need to be specified are nested two levels down within the property hierarchy, but the syntax of the \override command is quite simple:

```latex
e''2 \glissando b'
\once \override Glissando.bound-details.left.Y = #3
\once \override Glissando.bound-details.right.Y = #-2
e''2 \glissando b'
```

The units for the Y property are \texttt{staff-spaces}, with the center line of the staff being the zero point. For the glissando, this is the value for Y at the X-coordinate corresponding to the center point of each note head, if the line is imagined to be extended to there.
If \( Y \) is not set, the value is computed from the vertical position of the corresponding attachment point of the spanner.

In case of a line break, the values for the end points are specified by the `left-broken` and `right-broken` sub-lists of `bound-details`. For example:

\begin{verbatim}
\override Glissando.breakable = ##t
\override Glissando.bound-details.right-broken.Y = #-3
\end{verbatim}

A number of further properties of the `left` and `right` sub-lists of the `bound-details` property may be modified in the same way as \( Y \):

- **Y**: This 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.
  - For horizontal spanners, such as text spanners and trill spanners, it is hardcoded to 0.

- **attach-dir**: This 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. It is usually computed on the fly, and overriding it has little useful effect.

- **stencil**: Line spanners may have symbols at the beginning or end, which is contained in this sub-property. This is for internal use; it is recommended that `text` be used instead.

- **text**: This is a markup that is evaluated to yield the stencil. It is used to put cresc., tr and other text on horizontal spanners.

\begin{verbatim}
\override TextSpanner.bound-details.left.text
  = \markup { \small \bold Slower }
\end{verbatim}

- **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 either `stencil-align-dir-y` or `stencil-offset` will move the symbol at the edge vertically relative to the end point of the line:

\begin{verbatim}
\override TextSpanner.bound-details.left.stencil-align-dir-y = #-2
\end{verbatim}
\override TextSpanner.bound-details.right.stencil-align-dir-y = #UP
\override TextSpanner.bound-details.left.text = "ggg"
\override TextSpanner.bound-details.right.text = "hhh"

\relative { c'4 \startTextSpan c c c \stopTextSpan }

Note that negative values move the text up, contrary to the effect that might be expected, as a value of -1 or DOWN means align the bottom edge of the text with the spanner line. A value of 1 or UP aligns the top edge of the text with the spanner line.

arrow Setting this sub-property to #t produces an arrowhead at the end-points of the line.

padding This sub-property 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.

The music function \endSpanners terminates the spanner which starts on the immediately following note prematurely. It is terminated after exactly one note, or at the following bar line if to-barline is true and a bar line occurs before the next note.

\relative c' { \endSpanners c2 \startTextSpan c2 c2 \endSpanners c2 \< c2 c2 }

When using \endSpanners it is not necessary to close \startTextSpan with \stopTextSpan, nor is it necessary to close hairpins with \!.

See also

5.4.7 Visibility of objects
There are four main ways in which the visibility of layout objects can be controlled: their stencil can be removed, they can be made transparent, they can be colored white, or their break-visibility property can be overridden. The first three apply to all layout objects; the last to just a few – the breakable objects. The Learning Manual introduces these four techniques, see Section “Visibility and color of objects” in Learning Manual.

There are also a few other techniques which are specific to certain layout objects. These are covered under Special considerations.
Removing the stencil

Every layout object has a stencil property. By default this is set to the specific function which
draws that object. If this property is overridden to \#f no function will be called and the object
will not be drawn. The default action can be recovered with \revert.

\begin{verbatim}
\override Score.BarLine.stencil = ##f
\end{verbatim}

\begin{verbatim}
\revert Score.BarLine.stencil
\end{verbatim}

This rather common operation has a shortcut \omit:

\begin{verbatim}
\omit Score.BarLine
\end{verbatim}

\begin{verbatim}
\undo \omit Score.BarLine
\end{verbatim}

Making objects transparent

Every layout object has a transparent property which by default is set to \#f. If set to \#t the
object still occupies space but is made invisible.

\begin{verbatim}
\once \override NoteHead.transparent = ##t
\end{verbatim}

\begin{verbatim}
\hide NoteHead
\end{verbatim}

This rather common operation has a shortcut \hide:

\begin{verbatim}
\once \hide NoteHead
\end{verbatim}
Painting objects white

Every layout object has a color property which by default is set to black. If this is overridden to white the object will be indistinguishable from the white background. However, if the object crosses other objects the color of the crossing points will be determined by the order in which they are drawn, and this may leave a ghostly image of the white object, as shown here:

\override Staff.Clef.color = #white
\override Staff.Clef.layer = -1

This may be avoided by changing the order of printing the objects. All layout objects have a layer property which should be set to an integer. 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, although a few objects, including StaffSymbol and BarLine, are assigned a value of 0. The order of printing objects with the same value of layer is indeterminate.

In the example above the white clef, with a default layer value of 1, is drawn after the staff lines (default layer value 0), so overwriting them. To change this, the Clef object must be given in a lower value of layer, say -1, so that it is drawn earlier:

\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1

Using break-visibility

Most layout objects are printed only once, but some like bar lines, clefs, time signatures and key signatures, may need to be printed twice when a line break occurs – once at the end of the line and again at the start of the next line. Such objects are called breakable, and have a property, the break-visibility property to control their visibility at the three positions in which they may appear – at the start of a line, within a line if they are changed, and at the end of a line if a change takes place there.

For example, the time signature by default will be printed at the start of the first line, but nowhere else unless it changes, when it will be printed at the point at which the change occurs. If this change occurs at the end of a line the new time signature will be printed at the start of the next line and a cautionary time signature will be printed at the end of the previous line as well.

This behaviour is controlled by the break-visibility property, which is explained in Section “Visibility and color of objects” in Learning Manual. This property takes a vector of three booleans which, in order, determine whether the object is printed at the end of, within the body of, or at the beginning of a line. Or to be more precise, before a line break, where there is no line break, or after a line break.

Alternatively, these eight combinations may be specified by pre-defined functions, defined in scm/output-lib.scm, where the last three columns indicate whether the layout objects will be visible in the positions shown at the head of the columns:
### Function Vector Before At no After

<table>
<thead>
<tr>
<th>Function</th>
<th>Vector form</th>
<th>Before break</th>
<th>At no break</th>
<th>After break</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-visible</td>
<td>#(#t #t #t)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>begin-of-line-visible</td>
<td>#(#f #f #t)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>center-visible</td>
<td>#(#f #t #f)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>end-of-line-visible</td>
<td>#(#t #f #f)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>begin-of-line-invisible</td>
<td>#(#t #t #f)</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>center-invisible</td>
<td>#(#f #f #t)</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>end-of-line-invisible</td>
<td>#(#f #f #f)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>all-invisible</td>
<td>#(#f #f #f)</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

The default settings of `break-visibility` depend on the layout object. The following table shows all the layout objects of interest which are affected by `break-visibility` and the default setting of this property:

### Layout object Usual context Default setting

<table>
<thead>
<tr>
<th>Layout object</th>
<th>Usual context</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>BarLine</td>
<td>Score</td>
<td>calculated</td>
</tr>
<tr>
<td>BarNumber</td>
<td>Score</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>BreathingSign</td>
<td>Voice</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>Clef</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>Custos</td>
<td>Staff</td>
<td>end-of-line-visible</td>
</tr>
<tr>
<td>DoublePercentRepeat</td>
<td>Voice</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>KeyCancellation</td>
<td>Staff</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>KeySignature</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>ClefModifier</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>RehearsalMark</td>
<td>Score</td>
<td>end-of-line-invisible</td>
</tr>
<tr>
<td>TimeSignature</td>
<td>Staff</td>
<td>all-visible</td>
</tr>
</tbody>
</table>

The example below shows the use of the vector form to control the visibility of bar lines:

```latex
\relative {
  f'4 g a b
  f4 g a b
  % Remove bar line at the end of the current line
  \once \override Score.BarLine.break-visibility = #(#f #t #t)
  \break
  f4 g a b
  f4 g a b
}
```

Although all three components of the vector used to override `break-visibility` must be present, not all of them are effective with every layout object, and some combinations may even give errors. The following limitations apply:

- Bar lines cannot be printed at the start of line.
• A bar number cannot be printed at the start of the first line unless it is set to be different from 1.
• Clef – see the next section.
• Double percent repeats are either all printed or all suppressed. Use begin-of-line-invisible to print them and all-invisible to suppress them.
• Key signature – see the next section.
• ClefModifier – see the next section.

Special considerations

Visibility following explicit changes

The break-visibility property controls the visibility of key signatures and changes of clef only at the start of lines, i.e., after a break. It has no effect on the visibility of the key signature or clef following an explicit key change or an explicit clef change within or at the end of a line. In the following example the key signature following the explicit change to B-flat major is still visible, even though all-invisible is set.

\relative {
  \key g \major 
  f'4 g a b
  \% Try to remove all key signatures
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major 
  f4 g a b 
  \break 
  f4 g a b 
  f4 g a b 
}

The visibility of such explicit key signature and clef changes is controlled by the explicitKeySignatureVisibility and explicitClefVisibility properties. These are the equivalent of the break-visibility property and both take a vector of three booleans or the predefined functions listed above, exactly like break-visibility. Both are properties of the Staff context, not the layout objects themselves, and so they are set using the \set command. Both are set by default to all-visible. These properties control only the visibility of key signatures and clefs resulting from explicit changes and do not affect key signatures and clefs at the beginning of lines; break-visibility must still be overridden in the appropriate object to remove these.

\relative {
  \key g \major 
  f'4 g a b 
  \set Staff.explicitKeySignatureVisibility = #all-invisible
Visibility of cancelling accidentals

To remove the cancelling accidentals printed at an explicit key change, set the Staff context property printKeyCancellation to #f:

```latex
\relative { 
  \key g \major  
  f'4 g a b 
  \set Staff.explicitKeySignatureVisibility = #all-invisible  
  \set Staff.printKeyCancellation = ##f  
  \override Staff.KeySignature.break-visibility = #all-invisible  
  \key bes \major  
  f4 g a b \break  
  f4 g a b  
}
```

With these overrides only the accidentals before the notes remain to indicate the change of key.

Note that when changing the key to C major or A minor the cancelling accidentals would be the only indication of the key change. In this case setting printKeyCancellation to #f has no effect:

```latex
\relative { 
  \key g \major  
  f'4 g a b  
  \set Staff.explicitKeySignatureVisibility = #all-invisible  
  \set Staff.printKeyCancellation = ##f  
  \key c \major  
  f4 g a b \break  
```
To suppress the cancelling accidentals even when the key is changed to C major or A minor, override the visibility of the KeyCancellation grob instead:

\relative {
    \key g \major
    f'4 g a b
    \set Staff.explicitKeySignatureVisibility = #all-invisible
    \override Staff.KeyCancellation.break-visibility = #all-invisible
    \key c \major
    f4 g a b \break
    f4 g a b
    f4 g a b
}

**Automatic bars**

As a special case, the printing of bar lines can also be turned off by setting the automaticBars property in the Score context. If set to #f, bar lines will not be printed automatically; they must be explicitly created with a \bar command. Unlike the \cadenzaOn predefined command, measures are still counted. Bar generation will resume according to that count if this property is later set to #t. When set to #f, line breaks can occur only at explicit \bar commands.

**Transposed clefs**

The small transposition symbol on transposed clefs is produced by the ClefModifier layout object. Its visibility is automatically inherited from the Clef object, so it is not necessary to apply any required break-visibility overrides to the ClefModifier layout objects to suppress transposition symbols for invisible clefs.

For explicit clef changes, the explicitClefVisibility property controls both the clef symbol and any transposition symbol associated with it.

**See also**

5.4.8 Line styles

Some performance indications, e.g., *rallentando* and *accelerando* and *trills* are written as text and are extended over many measures with lines, sometimes dotted or wavy.

These all use the same routines as the glissando for drawing the texts and the lines, and tuning their behavior is therefore also done in the same way. It is done with a spanner, and the routine responsible for drawing the spanners is `ly:line-spanner::print`. This routine determines the exact location of the two *span points* and draws a line between them, in the style requested.

Here is an example showing the different line styles available, and how to tune them.

```latex
\relative { 
  d''2 \glissando d'2 
  \once \override Glissando.style = #'dashed-line 
  d,2 \glissando d'2 
  \override Glissando.style = #'dotted-line 
  d,2 \glissando d'2 
  \override Glissando.style = #'zigzag 
  d,2 \glissando d'2 
  \override Glissando.style = #'trill 
  d,2 \glissando d'2 }
```

The locations of the end-points of the spanner are computed on-the-fly for every graphic object, but it is possible to override these:

```latex
\relative { 
  e''2 \glissando f 
  \once \override Glissando.bound-details.right.Y = #-2 
  e2 \glissando f }
```

The value for $Y$ is set to $-2$ for the right end point. The left side may be similarly adjusted by specifying `left` instead of `right`.

If $Y$ is not set, the value is computed from the vertical position of the left and right attachment points of the spanner.

Other adjustments of spanners are possible, for details, see Section 5.4.6 [Spanners], page 637.

5.4.9 Rotating objects

Both layout objects and elements of markup text can be rotated by any angle about any point, but the method of doing so differs.
Rotating layout objects

All layout objects which support the \texttt{grob-interface} can be rotated by setting their \texttt{rotation} property. This takes a list of three items: the angle of rotation counter-clockwise, and the x and y coordinates of the point relative to the object’s reference point about which the rotation is to be performed. The angle of rotation is specified in degrees and the coordinates in staff-spaces.

The angle of rotation and the coordinates of the rotation point must be determined by trial and error.

There are only a few situations where the rotation of layout objects is useful; the following example shows one situation where they may be:

\begin{verbatim}
g4\< e' d'' f''\!
\override Hairpin.rotation = #'(15 -1 0)
g4\< e' d'' f''\!
\end{verbatim}

Rotating markup

All markup text can be rotated to lie at any angle by prefixing it with the \texttt{\rotate} command. The command takes two arguments: the angle of rotation in degrees counter-clockwise and the text to be rotated. The extents of the text are not rotated: they take their values from the extremes of the x and y coordinates of the rotated text. In the following example the \texttt{outside-staff-priority} property for text is set to \texttt{#f} to disable the automatic collision avoidance, which would push some of the text too high.

\begin{verbatim}
\override TextScript.outside-staff-priority = ##f
g4\markup { \rotate #30 "a G" }
b\markup { \rotate #30 "a B" }
des\markup { \rotate #30 "a D-Flat" }
fis\markup { \rotate #30 "an F-Sharp" }
\end{verbatim}

5.5 Advanced tweaks

This section discusses various approaches to fine tuning the appearance of the printed score.

See also

- Notation Reference: Section 5.2 [Explaining the Internals Reference], page 616, Section 5.3 [Modifying properties], page 619.
- Extending LilyPond: Section “Interfaces for programmers” in \textit{Extending}.
- Installed Files: \texttt{scm/define-grobs.scm}.
5.5.1 Aligning objects

Graphical objects which support the `self-alignment-interface` and/or the `side-position-interface` can be aligned to a previously placed object in a variety of ways. For a list of these objects, see Section “self-alignment-interface” in Internals Reference and Section “side-position-interface” in Internals Reference.

All graphical objects have a reference point, a horizontal extent and a vertical extent. The horizontal extent is a pair of numbers giving the displacements from the reference point of the left and right edges, displacements to the left being negative. The vertical extent is a pair of numbers giving the displacement from the reference point to the bottom and top edges, displacements down being negative.

An object’s position on a staff is given by the values of the `X-offset` and `Y-offset` properties. The value of `X-offset` gives the displacement from the X coordinate of the reference point of the parent object, and the value of `Y-offset` gives the displacement from the center line of the staff. The values of `X-offset` and `Y-offset` may be set directly or may be set to be calculated by procedures in order to achieve alignment with the parent object.

**Note:** Many objects have special positioning considerations which cause any setting of `X-offset` or `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface`. Overriding the `X-offset` or `Y-offset` properties to a fixed value causes the respective `self-alignment` property to be disregarded.

For example, an accidental can be repositioned vertically by setting `Y-offset` but any changes to `X-offset` have no effect.

Rehearsal marks may be aligned with breakable objects such as bar lines, clef symbols, time signature symbols and key signatures. There are special properties to be found in the `break-aligned-interface` for positioning rehearsal marks on such objects.

**See also**

Notation Reference: [Using the break-alignable-interface], page 652.

Extending LilyPond: Section “Callback functions” in Extending.

**Setting X-offset and Y-offset directly**

Numerical values may be given to the `X-offset` and `Y-offset` properties of many objects. The following example shows three notes with the default fingering position and the positions with `X-offset` and `Y-offset` modified.

```
a'-3
a'
\tweak X-offset #0
\tweak Y-offset #0
-3
a'
\tweak X-offset #-1
\tweak Y-offset #1
-3
```
Using the side-position-interface

An object which supports the side-position-interface can be placed next to its parent object so that the specified edges of the two objects touch. The object may be placed above, below, to the right or to the left of the parent. The parent cannot be specified; it is determined by the order of elements in the input stream. Most objects have the associated note head as their parent.

The values of the side-axis and direction properties determine where the object is to be placed, as follows:

<table>
<thead>
<tr>
<th>side-axis property</th>
<th>direction property</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>left</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>right</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>below</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>above</td>
</tr>
</tbody>
</table>

When side-axis is 0, X-offset should be set to the procedure ly:side-position-interface::x-aligned-side. This procedure will return the correct value of X-offset to place the object to the left or right side of the parent according to value of direction.

When side-axis is 1, Y-offset should be set to the procedure ly:side-position-interface::y-aligned-side. This procedure will return the correct value of Y-offset to place the object to the top or bottom of the parent according to value of direction.

Using the self-alignment-interface

Self-aligning objects horizontally

The horizontal alignment of an object which supports the self-alignment-interface is controlled by the value of the self-alignment-X property, provided the object’s X-offset property is set to ly:self-alignment-interface::x-aligned-on-self. self-alignment-X may be given any real value, in units of half the total X extent of the object. Negative values move the object to the right, positive to the left. A value of 0 centers the object on the reference point of its parent, a value of -1 aligns the left edge of the object on the reference point of its parent, and a value of 1 aligns the right edge of the object on the reference point of its parent. The symbols LEFT, CENTER, and RIGHT may be used instead of the values -1, 0, and 1, respectively.

Normally the \override command would be used to modify the value of self-alignment-X, but the \tweak command can be used to separately align several annotations on a single note:

```
\tweak self-alignment-X #-1
"left-aligned"
\tweak self-alignment-X #0
"center-aligned"
\tweak self-alignment-X #RIGHT
"right-aligned"
\tweak self-alignment-X #-2.5
"aligned further to the right"
```
Self-aligning objects vertically

Objects may be aligned vertically in an analogous way to aligning them horizontally if the Y-offset property is set to ly:self-alignment-interface::y-aligned-on-self. However, other mechanisms are often involved in vertical alignment: the value of Y-offset is just one variable taken into account. This may make adjusting the value of some objects tricky. The units are just half the vertical extent of the object, which is usually quite small, so quite large numbers may be required. A value of -1 aligns the lower edge of the object with the reference point of the parent object, a value of 0 aligns the center of the object with the reference point of the parent, and a value of 1 aligns the top edge of the object with the reference point of the parent. The symbols DOWN, CENTER, and UP may be substituted for -1, 0, and 1, respectively.

Self-aligning objects in both directions

By setting both X-offset and Y-offset, an object may be aligned in both directions simultaneously.

The following example shows how to adjust a fingering mark so that it nestles close to the note head.

a'  
\tweak self-alignment-X #0.5 % move horizontally left  
\tweak Y-offset #ly:self-alignment-interface::y-aligned-on-self  
\tweak self-alignment-Y #-1 % move vertically up  
3 % third finger

Using the break-alignable-interface

Rehearsal marks and bar numbers may be aligned with notation objects other than bar lines. These objects include ambitus, breathing-sign, clef, custos, staff-bar, left-edge, key-cancellation, key-signature, and time-signature.

Each type of object has its own default reference point, to which rehearsal marks are aligned:

% The rehearsal mark will be aligned to the right edge of the Clef  
\override Score.RehearsalMark.break-align-symbols = #'(clef)  
\key a \major  
\clef treble  
\mark "↓"  
e'1

% The rehearsal mark will be aligned to the left edge of the Time Signature  
\override Score.RehearsalMark.break-align-symbols = #'(time-signature)  
\key a \major  
\clef treble  
\time 3/4  
\mark "↓"  
e'2.

% The rehearsal mark will be centered above the Breath Mark  
\override Score.RehearsalMark.break-align-symbols = #'(breathing-sign)
A list of possible target alignment objects may be specified. If some of the objects are invisible at that point due to the setting of break-visibility or the explicit visibility settings for keys and clefs, the rehearsal mark or bar number is aligned to the first object in the list which is visible. If no objects in the list are visible the object is aligned to the bar line. If the bar line is invisible the object is aligned to the place where the bar line would be.

% The rehearsal mark will be aligned to the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "\"
e'1
% The rehearsal mark will be aligned to the right edge of the Clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef bass
\mark "\"
gis,1
% The rehearsal mark will be centered above the Bar Line
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "\"
e'1

The alignment of the rehearsal mark relative to the notation object can be changed, as shown in the following example. In a score with multiple staves, this setting should be done for all the staves.

% The RehearsalMark will be aligned with the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
The rehearsal mark can also be offset to the right or left of the left edge by an arbitrary amount. The units are staff-spaces:

\begin{music}
% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted right by 3.5 staff-spaces
\once \override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark "↓"
e'1
\end{music}

\begin{music}
% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted left by 2 staff-spaces
\once \override Score.KeySignature.break-align-anchor = #-2
\key a \major
\mark "↓"
e'1
\end{music}

5.5.2 Vertical grouping of grobs

The VerticalAlignment and VerticalAxisGroup grobs work together. VerticalAxisGroup groups together different grobs like Staff, Lyrics, etc. VerticalAlignment then vertically aligns the different grobs grouped together by VerticalAxisGroup. There is usually only one VerticalAlignment per score but every Staff, Lyrics, etc., has its own VerticalAxisGroup.

5.5.3 Modifying stencils

All layout objects have a stencil property which is part of the grob-interface. By default, this property is usually set to a function specific to the object that is tailor-made to render the symbol which represents it in the output. For example, the standard setting for the stencil property of the MultiMeasureRest object is ly:multi-measure-rest::print.

The standard symbol for any object can be replaced by modifying the stencil property to reference a different, specially-written, procedure. This requires a high level of knowledge of
the internal workings of LilyPond, but there is an easier way which can often produce adequate results.

This is to set the `stencil` property to the procedure which prints text – `ly:text-interface::print` – and to add a `text` property to the object which is set to contain the markup text which produces the required symbol. Due to the flexibility of markup, much can be achieved – see in particular [Graphic notation inside markup], page 262.

The following example demonstrates this by changing the note head symbol to a cross within a circle.

```latex
\begin{verbatim}
Xin0 = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
    \halign #-0.7 \draw-circle #0.85 #0.2 ##f
    \musicglyph "noteheads.s2cross"
  }
}
\relative {
  a' a \Xin0 a a
}
\end{verbatim}
```

Any of the `Feta` glyphs used in the Emmentaler font can be supplied to the \musicglyph markup command – see Section A.8 [The Emmentaler font], page 683.

EPS files and Postscript commands can both be inserted inline using the \epsfile and \postscript markup commands respectively – see Section A.11.3 [Graphic], page 736.

See also

Notation Reference: [Graphic notation inside markup], page 262, Section 1.8.2 [Formatting text], page 252, Section A.11 [Text markup commands], page 711, Section A.8 [The Emmentaler font], page 683, Section A.11.3 [Graphic], page 736.

5.5.4 Modifying shapes

Modifying ties and slurs

Ties, Slurs, PhrasingSlurs, LaissezVibrerTies and RepeatTies are all drawn as third-order Bézier curves. If the shape of the tie or slur which is calculated automatically is not optimum, the shape may be modified manually in two ways:

a. by specifying the displacements to be made to the control points of the automatically calculated Bézier curve, or

b. by explicitly specifying the positions of the four control points required to define the wanted curve.

Both methods are explained below. The first method is more suitable if only slight adjustments to the curve are required; the second may be better for creating curves which are related to just a single note.
Cubic Bézier curves

Third-order or cubic Bézier curves are defined by four control points. The first and fourth control points are precisely the starting and ending points of the curve. The intermediate two control points define the shape. Animations showing how the curve is drawn can be found on the web, but the following description may be helpful. The curve starts from the first control point heading directly towards the second, gradually bending over to head towards the third and continuing to bend over to head towards the fourth, arriving there travelling directly from the third control point. The curve is entirely contained in the quadrilateral defined by the four control points. Translations, rotations and scaling of the control points all result in exactly the same operations on the curve.

Specifying displacements from current control points

In this example the automatic placement of the tie is not optimum, and \tieDown would not help.

\begin{verbatim}
<<
 { e'1~ 1 } \\
 relative { r4 <g' c,> <g c,> <g c,> }
>>
\end{verbatim}

Adjusting the control points of the tie with \shape allows the collisions to be avoided.

The syntax of \shape is

\[-\] \shape displacements item

This will reposition the control-points of \textit{item} by the amounts given by \textit{displacements}. The \textit{displacements} argument is a list of number pairs or a list of such lists. Each element of a pair represents the displacement of one of the coordinates of a control-point. If \textit{item} is a string, the result is \once\override for the specified grob type. If \textit{item} is a music expression, the result is the same music expression with an appropriate \texttt{tweak} applied.

In other words, the \shape function can act as either a \once\override command or a \tweak command depending on whether the \textit{item} argument is a grob name, like “Slur”, or a music expression, like “\texttt{("}. The \textit{displacements} argument specifies the displacements of the four control points as a list of four pairs of (dx . dy) values in units of staff-spaces (or a list of such lists if the curve has more than one segment).

The leading hyphen is required if and only if the \tweak form is being used.

So, using the same example as above and the \once\override form of \shape, this will raise the tie by half a staff-space:

\begin{verbatim}
<<
 { 
 \shape #'(0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie 
 e'1~ 1 
 }
 \relative { r4 <g' c,> <g c,> <g c,> }
>>
\end{verbatim}
This positioning of the tie is better, but maybe it should be raised more in the center. The following example does this, this time using the alternative `\tweak` form:

```
<<
{ 
  e'1-\shape '((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e'
}
\relative { r4 <g' c,> <g c,> <g c,> }
>>
```

Changes to the horizontal positions of the control points may be made in the same way, and two different curves starting at the same musical moment may also be shaped:

```
\relative {
  c''8\( \ a\) a'4 e c\)
  \shape '((0.7 . -0.4) (0.5 . -0.4) (0.3 . -0.3) (0 . -0.2)) Slur
  \shape '((0 . 0) (0 . 0.5) (0 . 0.5) (0 . 0)) PhrasingSlur
  c8\( \ a\) a'4 e c\)
}
```

The `\shape` function can also displace the control points of curves which stretch across line breaks. Each piece of the broken curve can be given its own list of offsets. If changes to a particular segment are not needed, the empty list can serve as a placeholder. In this example the line break makes the single slur look like two:

```
\relative {
  c'4( f g c
  \break
  d,4 c' f, c)
}
```

Changing the shapes of the two halves of the slur makes it clearer that the slur continues over the line break:

```
% () may be used as a shorthand for ((0 . 0) (0 . 0) (0 . 0) (0 . 0))
```
% if any of the segments does not need to be changed
\relative c' {
  \shape #'((
    (0 . 0) (0 . 0) (0 . 0) (0 . 1))
    ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
  )
  Slur
c4( f g c
  \break
d,4 c' f, c)
}

If an S-shaped curve is required the control points must always be adjusted manually — LilyPond will never select such shapes automatically.

\relative c'' {
  c8( e b-> f d' a e-> g)
  \shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5))
  PhrasingSlur
c8( e b-> f d' a e-> g)
}

Specifying control points explicitly

The coordinates of the Bézier control points are specified in units of staff-spaces. The X coordinate is relative to the reference point of the note to which the tie or slur is attached, and the Y coordinate is relative to the staff center line. The coordinates are specified as a list of four pairs of decimal numbers (reals). One approach is to estimate the coordinates of the two end points, and then guess the two intermediate points. The optimum values are then found by trial and error. Be aware that these values may need to be manually adjusted if any further changes are made to the music or the layout.

One situation where specifying the control points explicitly is preferable to specifying displacements is when they need to be specified relative to a single note. Here is an example of this. It shows one way of indicating a slur extending into alternative sections of a volta repeat.

\relative c''1
\repeat volta 3 { c4 d( e f }
\alternative {
  { g2) d }
  {
    g2
    \% create a slur and move it to a new position
}
% the <> is just an empty chord to carry the slur termination
-\tweak control-points #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <>)
f,
}
{e'2
% create a slur and move it to a new position
-\tweak control-points #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <>)
f,
}

Known issues and warnings
It is not possible to modify shapes of ties or slurs by changing the control-points property if there are multiple ties or slurs at the same musical moment – the \tweak command will also not work in this case. However, the tie-configuration property of TieColumn can be overridden to set start line and direction as required.

See also
Internals Reference: Section “TieColumn” in Internals Reference.

5.5.5 Modifying broken spanners

Using \alterBroken
When a spanner crosses a line break or breaks, each piece inherits the attributes of the original spanner. Thus, ordinary tweaking of a broken spanner applies the same modifications to each of its segments. In the example below, overriding thickness affects the slur on either side of the line break.

\relative c'' {
  r2
  \once\override Slur.thickness = 10
c8( d e f
  \break
g8 f e d) r2
}

\relative c' {
  r2
  \once\override Slur.thickness = 10
c8( d e f
  \break
g8 f e d) r2
}
Independently modifying the appearance of individual pieces of a broken spanner is possible with the `\alterBroken` command. This command can produce either an `\override` or a `\tweak` of a spanner property.

The syntax for `\alterBroken` is

```
[-]\alterBroken property values item
```

The argument `values` is a list of values, one for each broken piece. If `item` is a grob name like `Slur` or `Staff.PianoPedalBracket`, the result is an `\override` of the specified grob type. If `item` is a music expression such as `("` or `["] the result is the same music expression with an appropriate tweak applied.

The leading hyphen must be used with the `\tweak` form. Do not add it when `\alterBroken` is used as an `\override`.

In its `\override` usage, `\alterBroken` may be prefaced by `\once` or `\temporary` and reverted by using `\revert` with `property` (see Section “Intermediate substitution functions” in Extending).

The following code applies an independent `\override` to each of the slur segments in the previous example:

```
\relative c'' {
  r2
  \alterBroken thickness #'(10 1) Slur
c8( d e f
  \break
g8 f e d) r2
}
```

The `\alterBroken` command may be used with any spanner object, including `Tie`, `PhrasingSlur`, `Beam` and `TextSpanner`. For example, an editor preparing a scholarly edition may wish to indicate the absence of part of a phrasing slur in a source by dashing only the segment which has been added. The following example illustrates how this can be done, in this case using the `\tweak` form of the command:

```
% The empty list is conveniently used below, because it is the
% default setting of dash-definition, resulting in a solid curve.
\relative {
  c'2-\alterBroken dash-definition #'((() ((0 1.0 0.4 0.75))) \(e
  \break
g2 e\)
}
```

It is important to understand that `\alterBroken` will set each piece of a broken spanner to the corresponding value in `values`. When there are fewer values than pieces, any additional piece will be assigned the empty list. This may lead to undesired results if the layout property is not set to the empty list by default. In such cases, each segment should be assigned an appropriate value.

**Known issues and warnings**

Line breaks may occur in different places following changes in layout. Settings chosen for `\alterBroken` may be unsuitable for a spanner that is no longer broken or is split into more segments than before. Explicit use of `\break` can guard against this situation.

The `\alterBroken` command is ineffective for spanner properties accessed before line-breaking such as `direction`.

**See also**

Extending LilyPond: Section “Difficult tweaks” in *Extending*.

### 5.5.6 Unpure-pure containers

Unpure-pure containers are useful for overriding `Y-axis` spacing calculations - specifically `Y-offset` and `Y-extent` - with a Scheme function instead of a literal (i.e., a number or pair).

For certain grobs, the `Y-extent` is based on the `stencil` property, overriding the stencil property of one of these will require an additional `Y-extent` override with an unpure-pure container. When a function overrides a `Y-offset` and/or `Y-extent` it is assumed that this will trigger line breaking calculations too early during compilation. So the function is not evaluated at all (usually returning a value of ‘0’ or ‘(0 . 0)’) which can result in collisions. A ‘pure’ function will not affect properties, objects or grob suicides and therefore will always have its `Y-axis-related` evaluated correctly.

Currently, there are about thirty functions that are already considered ‘pure’ and Unpure-pure containers are a way to set functions not on this list as ‘pure’. The ‘pure’ function is evaluated before any line-breaking and so the horizontal spacing can be adjusted ‘in time’. The ‘unpure’ function is then evaluated after line breaking.

**Note:** As it is difficult to always know which functions are on this list we recommend that any ‘pure’ functions you create do not use `Beam` or `VerticalAlignment` grobs.

An unpure-pure container is constructed as follows;

```
(ly:make-unpure-pure-container f0 f1)
```

where `f0` is a function taking `n` arguments (`n >= 1`) and the first argument must always be the grob. This is the function that gives the actual result. `f1` is the function being labeled as ‘pure’ that takes `n + 2` arguments. Again, the first argument must always still be the grob but the second and third are ‘start’ and ‘end’ arguments.

`start` and `end` are, for all intents and purposes, dummy values that only matter for `Spanners` (i.e `Hairpin` or `Beam`), that can return different height estimations based on a starting and ending column.

The rest are the other arguments to the first function (which may be none if `n = 1`).

The results of the second function are used as an approximation of the value needed which is then used by the first function to get the real value which is then used for fine-tuning much later during the spacing process.

```
#(define (square-line-circle-space grob)
```
(let* ((pitch (ly:event-property (ly:grob-property grob 'cause) 'pitch))
        (notename (ly:pitch-notename pitch)))
    (if (= 0 (modulo notename 2))
        (make-circle-stencil 0.5 0.0 #t)
        (make-filled-box-stencil '(0 . 1.0)
                       '(-0.5 . 0.5))))

squareLineCircleSpace = {
    \override NoteHead.stencil = #square-line-circle-space
}

smartSquareLineCircleSpace = {
    \squareLineCircleSpace
    \override NoteHead.Y-extent = #(ly:make-unpure-pure-container
        ly:grob::stencil-height
        (lambda (grob start end) (ly:grob::stencil-height grob)))
}

\new Voice \with { \remove "Stem_engraver" } \relative c' {
    \squareLineCircleSpace
    cis4 ces disis d
    \smartSquareLineCircleSpace
    cis4 ces disis d
}

\note
\foreign{\textcopyright}

In the first measure, without the unpure-pure container, the spacing engine does not know
the width of the note head and lets it collide with the accidentals. In the second measure, with
unpure-pure containers, the spacing engine knows the width of the note heads and avoids the
collision by lengthening the line accordingly.

Usually for simple calculations nearly-identical functions for both the ‘unpure’ and ‘pure’
parts can be used, by only changing the number of arguments passed to, and the scope of, the
function. This use case is frequent enough that \texttt{ly:make-unpure-pure-container}
constructs such a second function by default when called with only one function argument.

\begin{quote}
\textbf{Note:} If a function is labeled as ‘pure’ and it turns out not to be, the
results can be unexpected.
\end{quote}

5.6 Using music functions

Where tweaks need to be reused with different music expressions, it is often convenient to make
the tweak part of a \textit{music function}. In this section, we discuss only \textit{substitution}
functions, where the object is to substitute a variable into a piece of LilyPond input code. Other more complex
functions are described in Section “Music functions” in \textit{Extending}. 
5.6.1 Substitution function syntax

Making a function that substitutes a variable into LilyPond code is easy. The general form of these functions is

```lily
function =
#(define-music-function
 (arg1 arg2 ...)
 (type1? type2? ...)
#{
 ...music... 
 #})
```

where

- `argN` is the `n`th argument.
- `typeN?` is a scheme type predicate for which `argN` must return `#t`.
- `...music...` is normal LilyPond input, using `$` (in places where only LilyPond constructs are allowed) or `#` (to use it as a Scheme value or music function argument or music inside of music lists) to reference arguments (e.g., `'#arg1').

The list of type predicates is required. Some of the most common type predicates used in music functions are:

- `boolean?`
- `cheap-list?` (use instead of ‘list?’ for faster processing)
- `ly:duration?`
- `ly:music?`
- `ly:pitch?`
- `markup?`
- `number?`
- `pair?`
- `string?`
- `symbol?`

For a list of available type predicates, see Section A.21 [Predefined type predicates], page 822. User-defined type predicates are also allowed.

See also

- Notation Reference: Section A.21 [Predefined type predicates], page 822.
- Extending LilyPond: Section “Music functions” in Extending.
- Installed Files: lily/music-scheme.cc, scm/c++.scm, scm/lily.scm.

5.6.2 Substitution function examples

This section introduces some substitution function examples. These are not intended to be exhaustive, but rather to demonstrate some of the possibilities of simple substitution functions.

In the first example, a function is defined that simplifies setting the padding of a TextScript:

```lily
padText =
#(define-music-function
 (padding)
 (number?)
#{
 \once \override TextScript.padding = #padding
```
In addition to numbers, we can use music expressions such as notes for arguments to music functions:

```latex
\begin{verbatim}
customNote =
#(define-music-function
 (note)
 (ly:music?)
#{
 \tweak NoteHead.stencil #ly:text-interface::print
 \tweak NoteHead.text
 \markup \musicglyph "custodes.mensural.u0"
 \tweak Stem.stencil ##f
 #note
 #})
\end{verbatim}
```

Both of those functions are simple single expressions where only the last element of a function call or override is missing. For those particular function definitions, there is a simpler alternative syntax, namely just writing out the constant part of the expression and replacing its final missing element with \etc:

```latex
\begin{verbatim}
padText =
 \once \override TextScript.padding = \etc
\end{verbatim}
```

```
\relative { c'4 "piu mosso" b a b
 \padText #1.8
 c4 "piu mosso" b a b
 \padText #2.6
 c4 "piu mosso" b a b
}
```

```
\begin{verbatim}
\relative { c'4 d e f \customNote g }
\end{verbatim}
```

```
\begin{verbatim}
\relative { c'4 "piu mosso" b a b
 \padText #1.8
 c4 "piu mosso" b a b
 \padText #2.6
 c4 "piu mosso" b a b
}
```

```
Substitution functions with multiple arguments can be defined:

```plaintext
tempoPadded =
  #(define-music-function
   (padding tempotext)
   (number? markup?)
   #{
     \once \override Score.MetronomeMark.padding = #padding
     \tempo \markup { \bold #tempotext }
   })
```

```plaintext\relative { \tempo \markup { "Low tempo" }
c''4 d e f g1
\tempoPadded #4.0 "High tempo"
g4 f e d c1 }
```

Low tempo

High tempo
Appendix A Notation manual tables

A.1 Chord name chart

The following chart shows LilyPond’s standard system for printing chord names, along with the pitches they represent. Additional (unsupported) naming systems are also demonstrated in the “Chord names alternative” snippet in Section “Chords” in Snippets, including the notation inspired by Harald Banter (1982) that was used by default in early LilyPond releases (up to version 1.7).
### A.2 Common chord modifiers

The following table shows chord modifiers that can be used to generate standard chord structures.

<table>
<thead>
<tr>
<th>Type</th>
<th>Interval</th>
<th>Modifier</th>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Major third, perfect fifth</td>
<td>5 or nothing</td>
<td>c1:5</td>
<td></td>
</tr>
<tr>
<td>Minor</td>
<td>Minor third, perfect fifth</td>
<td>m or m5</td>
<td>c1:m</td>
<td></td>
</tr>
<tr>
<td>Augmented</td>
<td>Major third, augmented fifth</td>
<td>aug</td>
<td>c1:aug</td>
<td></td>
</tr>
<tr>
<td>Diminished</td>
<td>Minor third, diminished fifth</td>
<td>dim</td>
<td>c1:dim</td>
<td></td>
</tr>
<tr>
<td>Dominant seventh</td>
<td>Major triad, minor seventh</td>
<td>7</td>
<td>c1:7</td>
<td></td>
</tr>
<tr>
<td>Major seventh</td>
<td>Major triad, major seventh</td>
<td>maj7 or maj</td>
<td>c1:maj7</td>
<td></td>
</tr>
<tr>
<td>Minor seventh</td>
<td>Minor triad, minor seventh</td>
<td>m7</td>
<td>c1:m7</td>
<td></td>
</tr>
<tr>
<td>Diminished seventh</td>
<td>Diminished triad, diminished seventh</td>
<td>dim7</td>
<td>c1:dim7</td>
<td></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td>Symbol</td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Augmented seventh</td>
<td>Augmented triad, minor seventh</td>
<td>$\text{aug7}$</td>
<td>$c_1: \text{aug7}$</td>
<td></td>
</tr>
<tr>
<td>Half-diminished seventh</td>
<td>Diminished triad, minor seventh</td>
<td>$m7.5^-$</td>
<td>$c_1:m7.5^-$</td>
<td></td>
</tr>
<tr>
<td>Minor-major seventh</td>
<td>Minor triad, major seventh</td>
<td>$m7^+$</td>
<td>$c_1:m7^+$</td>
<td></td>
</tr>
<tr>
<td>Major sixth</td>
<td>Major triad, sixth</td>
<td>$6$</td>
<td>$c_1:6$</td>
<td></td>
</tr>
<tr>
<td>Minor sixth</td>
<td>Minor triad, sixth</td>
<td>$m6$</td>
<td>$c_1:m6$</td>
<td></td>
</tr>
<tr>
<td>Dominant ninth</td>
<td>Dominant seventh, major ninth</td>
<td>$9$</td>
<td>$c_1:9$</td>
<td></td>
</tr>
<tr>
<td>Major ninth</td>
<td>Major seventh, major ninth</td>
<td>$\text{maj9}$</td>
<td>$c_1:\text{maj9}$</td>
<td></td>
</tr>
<tr>
<td>Minor ninth</td>
<td>Minor seventh, major ninth</td>
<td>$m9$</td>
<td>$c_1:m9$</td>
<td></td>
</tr>
<tr>
<td>Dominant eleventh</td>
<td>Dominant ninth, perfect eleventh</td>
<td>$11$</td>
<td>$c_1:11$</td>
<td></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td>Major/Natural</td>
<td>Minor</td>
<td>Dominant</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------</td>
<td>-----------</td>
</tr>
<tr>
<td>maj11</td>
<td>Major ninth,</td>
<td>c1:maj11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>perfect eleventh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m11</td>
<td>Minor ninth,</td>
<td>c1:m11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>perfect eleventh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Dominant ninth,</td>
<td>c1:13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>major thirteenth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.11</td>
<td>Dominant eleventh,</td>
<td>c1:13.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>major thirteenth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maj13.11</td>
<td>Major eleventh,</td>
<td>c1:maj13.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>major thirteenth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m13.11</td>
<td>Minor eleventh,</td>
<td>c1:m13.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>major thirteenth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sus2</td>
<td>Major second,</td>
<td>c1:sus2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>perfect fifth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sus4</td>
<td>Perfect fourth,</td>
<td>c1:sus4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>perfect fifth</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

Power chord
(two-voiced) Perfect fifth 1.5 c1:5

Power chord
(three-voiced) Perfect fifth, octave 1.5.8 c1:5.8

A.3 Predefined string tunings
The chart below shows the predefined string tunings.

Guitar tunings

guitar-tuning
guitar-seven-string-tuning
guitar-drop-d-tuning

4
guitar-drop-c-tuning
guitar-open-g-tuning
guitar-open-d-tuning

7
guitar-dadgad-tuning
guitar-lute-tuning
guitar-asus4-tuning

Bass tunings

bass-tuning

bass-four-string-tuning

bass-drop-d-tuning

13
bass-five-string-tuning

bass-six-string-tuning

Mandolin tunings

mandolin-tuning
A.4 Predefined fretboard diagrams

Diagrams for Guitar

\[\text{C} \quad \text{Cm} \quad \text{C}+ \quad \text{C}^\circ \quad \text{C}^7 \quad \text{C}^\Delta \quad \text{Cm}^7 \quad \text{C}^9\]

\[\text{C}# \quad \text{C}^\#m \quad \text{C}#+ \quad \text{C}^\#^\circ \quad \text{C}^\#7 \quad \text{C}^\#^\Delta \quad \text{C}^\#m^7 \quad \text{C}^\#^9\]

\[\text{Db} \quad \text{Dbm} \quad \text{Db}+ \quad \text{Db}^\circ \quad \text{Db}^7 \quad \text{Db}^\Delta \quad \text{Dbm}^7 \quad \text{Db}^9\]

\[\text{D} \quad \text{Dm} \quad \text{D}+ \quad \text{D}^\circ \quad \text{D}^7 \quad \text{D}^\Delta \quad \text{Dm}^7 \quad \text{D}^9\]
### Appendix A: Notation manual tables

| Key | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D#  | D#m | D#+ | D#o | D#o7 | D#7 | D#Δ | D#m7 | D#9 | E♭  | E♭m | E♭+ | E♭o |
| 3121| 3214| 321 | 3141| 1324 | 1324| 1324| 1234 | 21333|
| E♭m | E♭+ | E♭ō | E♭ō7 | E♭7 | E♭Δ | E♭m7 | E♭9 | E♭m | E♭+ | E♭ō | E♭ō7 | E♭7 |
| 3121| 3214| 321 | 3141| 1324 | 1324| 1324| 1234 | 21333|
| E   | Em  | E+  | Eō  | Eō7 | E7  | EΔ  | Em7  | E9  | F   | Fm  | F+  | Fō  |
| 231 | 231 | 321 | 3141| 1324 | 1324| 1324| 13111| 13114|
| F#  | F#m | F#+ | F#o | F#o7 | F#7 | F#Δ | F#m7 | F#9 | Gb  | Gbm | Gb+ | Gbō |
| 1321| 13411| 2143| 3141| 1324 | 1324| 1324| 13111| 13114|
| Gbō | Gb7  | GbΔ | Gbm7 | Gb9 | Gbō | Gb7  | GbΔ | Gbm7 | Gb9 | G    | Gm  | G+  |
| 1321| 13411| 2143| 3141| 1324 | 1324| 1324| 13111| 13114|
| G#  | G#m | G#+ | G#o | G#o7 | G#7 | G#Δ | G#m7 | G#9 | Ab  | Abm | Ab+ | Abō |
| 13421| 13411| 4312| 3141| 1324 | 1324| 1324| 13111| 13114|
| Abō | Ab7  | AbΔ | Abm7 | Ab9 | Abō | Ab7  | AbΔ | Abm7 | Ab9 | A    | Am  | A+  |
| 13421| 13411| 4312| 3141| 1324 | 1324| 1324| 13111| 13114|
| A   | Am  | A+  | Aō  | Aō7 | A7  | AΔ  | Am7  | A9  | | | | | |
### Appendix A: Notation manual tables

#### Diagrams for Ukulele

<table>
<thead>
<tr>
<th>C</th>
<th>Cm</th>
<th>C+</th>
<th>C°</th>
<th>C7</th>
<th>CΔ</th>
<th>Cm7</th>
<th>C6</th>
<th>C7</th>
<th>C°</th>
<th>Cm7</th>
<th>C6</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
<td><img src="image9.png" alt="Diagram" /></td>
<td><img src="image10.png" alt="Diagram" /></td>
<td><img src="image11.png" alt="Diagram" /></td>
<td><img src="image12.png" alt="Diagram" /></td>
<td><img src="image13.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C#</th>
<th>C#m</th>
<th>C#+</th>
<th>C#°</th>
<th>C#7</th>
<th>C#Δ</th>
<th>C#m7</th>
<th>C#6</th>
<th>C#7</th>
<th>C#°</th>
<th>C#m7</th>
<th>C#6</th>
<th>C#9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image14.png" alt="Diagram" /></td>
<td><img src="image15.png" alt="Diagram" /></td>
<td><img src="image16.png" alt="Diagram" /></td>
<td><img src="image17.png" alt="Diagram" /></td>
<td><img src="image18.png" alt="Diagram" /></td>
<td><img src="image19.png" alt="Diagram" /></td>
<td><img src="image20.png" alt="Diagram" /></td>
<td><img src="image21.png" alt="Diagram" /></td>
<td><img src="image22.png" alt="Diagram" /></td>
<td><img src="image23.png" alt="Diagram" /></td>
<td><img src="image24.png" alt="Diagram" /></td>
<td><img src="image25.png" alt="Diagram" /></td>
<td><img src="image26.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Db</th>
<th>Dbm</th>
<th>Db+</th>
<th>Db°</th>
<th>Db7</th>
<th>DbΔ</th>
<th>Dbm7</th>
<th>Db6</th>
<th>Db7</th>
<th>Db°</th>
<th>Dbm7</th>
<th>Db6</th>
<th>Db9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image27.png" alt="Diagram" /></td>
<td><img src="image28.png" alt="Diagram" /></td>
<td><img src="image29.png" alt="Diagram" /></td>
<td><img src="image30.png" alt="Diagram" /></td>
<td><img src="image31.png" alt="Diagram" /></td>
<td><img src="image32.png" alt="Diagram" /></td>
<td><img src="image33.png" alt="Diagram" /></td>
<td><img src="image34.png" alt="Diagram" /></td>
<td><img src="image35.png" alt="Diagram" /></td>
<td><img src="image36.png" alt="Diagram" /></td>
<td><img src="image37.png" alt="Diagram" /></td>
<td><img src="image38.png" alt="Diagram" /></td>
<td><img src="image39.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>Dm</th>
<th>D+</th>
<th>D°</th>
<th>D7</th>
<th>DΔ</th>
<th>Dm7</th>
<th>D6</th>
<th>D7</th>
<th>D°</th>
<th>Dm7</th>
<th>D6</th>
<th>D9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image40.png" alt="Diagram" /></td>
<td><img src="image41.png" alt="Diagram" /></td>
<td><img src="image42.png" alt="Diagram" /></td>
<td><img src="image43.png" alt="Diagram" /></td>
<td><img src="image44.png" alt="Diagram" /></td>
<td><img src="image45.png" alt="Diagram" /></td>
<td><img src="image46.png" alt="Diagram" /></td>
<td><img src="image47.png" alt="Diagram" /></td>
<td><img src="image48.png" alt="Diagram" /></td>
<td><img src="image49.png" alt="Diagram" /></td>
<td><img src="image50.png" alt="Diagram" /></td>
<td><img src="image51.png" alt="Diagram" /></td>
<td><img src="image52.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D#</th>
<th>D#m</th>
<th>D#+</th>
<th>D#°</th>
<th>D#7</th>
<th>D#Δ</th>
<th>D#m7</th>
<th>D#6</th>
<th>D#7</th>
<th>D#°</th>
<th>D#m7</th>
<th>D#6</th>
<th>D#9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image53.png" alt="Diagram" /></td>
<td><img src="image54.png" alt="Diagram" /></td>
<td><img src="image55.png" alt="Diagram" /></td>
<td><img src="image56.png" alt="Diagram" /></td>
<td><img src="image57.png" alt="Diagram" /></td>
<td><img src="image58.png" alt="Diagram" /></td>
<td><img src="image59.png" alt="Diagram" /></td>
<td><img src="image60.png" alt="Diagram" /></td>
<td><img src="image61.png" alt="Diagram" /></td>
<td><img src="image62.png" alt="Diagram" /></td>
<td><img src="image63.png" alt="Diagram" /></td>
<td><img src="image64.png" alt="Diagram" /></td>
<td><img src="image65.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eb</th>
<th>Ebm</th>
<th>Eb+</th>
<th>Eb°</th>
<th>Eb7</th>
<th>EbΔ</th>
<th>Ebm7</th>
<th>Eb6</th>
<th>Eb7</th>
<th>Eb°</th>
<th>Ebm7</th>
<th>Eb6</th>
<th>Eb9</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image66.png" alt="Diagram" /></td>
<td><img src="image67.png" alt="Diagram" /></td>
<td><img src="image68.png" alt="Diagram" /></td>
<td><img src="image69.png" alt="Diagram" /></td>
<td><img src="image70.png" alt="Diagram" /></td>
<td><img src="image71.png" alt="Diagram" /></td>
<td><img src="image72.png" alt="Diagram" /></td>
<td><img src="image73.png" alt="Diagram" /></td>
<td><img src="image74.png" alt="Diagram" /></td>
<td><img src="image75.png" alt="Diagram" /></td>
<td><img src="image76.png" alt="Diagram" /></td>
<td><img src="image77.png" alt="Diagram" /></td>
<td><img src="image78.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Em</td>
<td>E+</td>
<td>E°</td>
<td>E^</td>
<td>E^</td>
<td>Em^</td>
<td>E^</td>
<td>E^</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

| F  | Fm | F+ | F° | F\^ | F\^ | F\^ | F\^ | F\^ |
| 1  | 1  | 1  | 2  | 4  | 4  | 1  | 1  | 1  |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

| Gb | Gbm | Gb+ | Gb° | Gb\^ | Gb\^ | Gb\^ | Gb\^ | Gb\^ |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

| G  | Gm | G+ | G° | G\^ | G\^ | G\^ | G\^ | G\^ |
| 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |
| 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |

| Ab | Abm | Ab+ | Ab° | Ab\^ | Ab\^ | Ab\^ | Ab\^ | Ab\^ |
| 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  |
| 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  |

| A  | Am | A+ | A° | A\^ | A\^ | A\^ | A\^ | A\^ |
| 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |
| 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |

| A\# | A\#m | A\#+ | A\#° | A\#\^ | A\#\^ | A\#\^ | A\#\^ | A\#\^ |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

| Bb | Bbm | Bb+ | Bb° | Bb\^ | Bb\^ | Bb\^ | Bb\^ | Bb\^ |
| 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  |
| 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  |
Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>Cm</th>
<th>C+</th>
<th>C^7</th>
<th>C^Δ</th>
<th>Cm^7</th>
<th>C^o</th>
<th>C^6</th>
<th>C^sus2</th>
<th>C^sus4</th>
<th>C^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Bm</td>
<td>B+</td>
<td>B^o</td>
<td>B^7</td>
<td>B^Δ</td>
<td>Bm^7</td>
<td>B^6</td>
<td>B^sus2</td>
<td>B^sus4</td>
<td>B^9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3211</td>
<td>3111</td>
<td>231</td>
<td>412</td>
<td>2111</td>
<td>2211</td>
<td>1111</td>
<td>1423</td>
<td>4132</td>
<td>2313</td>
<td>1324</td>
</tr>
</tbody>
</table>

Diagrams for Mandolin
A.5 Predefined paper sizes

Paper sizes are defined in $\text{scm/paper.scm}$

The “ISO 216” A Series

"a10" (26 x 37 mm)
"a9" (37 x 52 mm)
"a8" (52 x 74 mm)
"a7" (74 x 105 mm)
"a6" (105 x 148 mm)
"a5" (148 x 210 mm)
"a4" (210 x 297 mm)
"a3" (297 x 420 mm)
"a2" (420 x 594 mm)
"a1" (594 x 841 mm)
"a0" (841 x 1189 mm)

The “ISO 216” B Series

"b10" (31 x 44 mm)
"b9" (44 x 62 mm)
"b8" (62 x 88 mm)
"b7" (88 x 125 mm)
"b6" (125 x 176 mm)
"b5" (176 x 250 mm)
"b4" (250 x 353 mm)
"b3" (353 x 500 mm)
"b2" (500 x 707 mm)
"b1" (707 x 1000 mm)
"b0" (1000 x 1414 mm)

Two extended sizes as defined in “DIN 476”

"4a0" (1682 x 2378 mm)
"2a0" (1189 x 1682 mm)

“ISO 269” standard C series

"c10" (28 x 40 mm)
"c9" (40 x 57 mm)
"c8" (57 x 81 mm)
"c7" (81 x 114 mm)
"c6" (114 x 162 mm)
"c5" (162 x 229 mm)
"c4" (229 x 324 mm)
"c3" (324 x 458 mm)
"c2" (458 x 648 mm)
"c1" (648 x 917 mm)
"c0" (917 x 1297 mm)

North American paper sizes
"junior-legal" (8.0 x 5.0 in)
"legal" (8.5 x 14.0 in)
"ledger" (17.0 x 11.0 in)
"letter" (8.5 x 11.0 in)
"tabloid" (11.0 x 17.0 in)
"11x17" (11.0 x 17.0 in)
"17x11" (17.0 x 11.0 in)

Government-letter by IEEE Printer Working Group, for children’s writing
"government-letter" (8 x 10.5 in)
"government-legal" (8.5 x 13.0 in)
"philippine-legal" (8.5 x 13.0 in)

ANSI sizes
"ansi a" (8.5 x 11.0 in)
"ansi b" (17.0 x 11.0 in)
"ansi c" (17.0 x 22.0 in)
"ansi d" (22.0 x 34.0 in)
"ansi e" (34.0 x 44.0 in)
"engineering f" (28.0 x 40.0 in)

North American Architectural sizes
"arch a" (9.0 x 12.0 in)
"arch b" (12.0 x 18.0 in)
"arch c" (18.0 x 24.0 in)
"arch d" (24.0 x 36.0 in)
"arch e" (36.0 x 48.0 in)
"arch e1" (30.0 x 42.0 in)
Antique sizes still used in the United Kingdom

<table>
<thead>
<tr>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;statement&quot;</td>
<td>(5.5 x 8.5 in)</td>
</tr>
<tr>
<td>&quot;half letter&quot;</td>
<td>(5.5 x 8.5 in)</td>
</tr>
<tr>
<td>&quot;quarto&quot;</td>
<td>(8.0 x 10.0 in)</td>
</tr>
<tr>
<td>&quot;octavo&quot;</td>
<td>(6.75 x 10.5 in)</td>
</tr>
<tr>
<td>&quot;executive&quot;</td>
<td>(7.25 x 10.5 in)</td>
</tr>
<tr>
<td>&quot;monarch&quot;</td>
<td>(7.25 x 10.5 in)</td>
</tr>
<tr>
<td>&quot;foolscap&quot;</td>
<td>(8.27 x 13.0 in)</td>
</tr>
<tr>
<td>&quot;folio&quot;</td>
<td>(8.27 x 13.0 in)</td>
</tr>
<tr>
<td>&quot;super-b&quot;</td>
<td>(13.0 x 19.0 in)</td>
</tr>
<tr>
<td>&quot;post&quot;</td>
<td>(15.5 x 19.5 in)</td>
</tr>
<tr>
<td>&quot;crown&quot;</td>
<td>(15.0 x 20.0 in)</td>
</tr>
<tr>
<td>&quot;large post&quot;</td>
<td>(16.5 x 21.0 in)</td>
</tr>
<tr>
<td>&quot;demy&quot;</td>
<td>(17.5 x 22.5 in)</td>
</tr>
<tr>
<td>&quot;medium&quot;</td>
<td>(18.0 x 23.0 in)</td>
</tr>
<tr>
<td>&quot;broadsheet&quot;</td>
<td>(18.0 x 24.0 in)</td>
</tr>
<tr>
<td>&quot;royal&quot;</td>
<td>(20.0 x 25.0 in)</td>
</tr>
<tr>
<td>&quot;elephant&quot;</td>
<td>(23.0 x 28.0 in)</td>
</tr>
<tr>
<td>&quot;double demy&quot;</td>
<td>(22.5 x 35.0 in)</td>
</tr>
<tr>
<td>&quot;quad demy&quot;</td>
<td>(35.0 x 45.0 in)</td>
</tr>
<tr>
<td>&quot;atlas&quot;</td>
<td>(26.0 x 34.0 in)</td>
</tr>
<tr>
<td>&quot;imperial&quot;</td>
<td>(22.0 x 30.0 in)</td>
</tr>
<tr>
<td>&quot;antiquarian&quot;</td>
<td>(31.0 x 53.0 in)</td>
</tr>
</tbody>
</table>

PA4 based sizes

<table>
<thead>
<tr>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;pa0&quot;</td>
<td>(840 x 1120 mm)</td>
</tr>
<tr>
<td>&quot;pa1&quot;</td>
<td>(560 x 840 mm)</td>
</tr>
<tr>
<td>&quot;pa2&quot;</td>
<td>(420 x 560 mm)</td>
</tr>
</tbody>
</table>
### A.6 MIDI instruments

The following is a list of names that can be used for the `midiInstrument` property. The order of the instruments below, starting in the left-hand column moving down, corresponds to the General MIDI Standard’s 128 Program Numbers.

<table>
<thead>
<tr>
<th>Instrument 1</th>
<th>Instrument 2</th>
<th>Instrument 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>acoustic grand</td>
<td>contrabass</td>
<td>lead 7 (fifths)</td>
</tr>
<tr>
<td>bright acoustic</td>
<td>tremolo strings</td>
<td>lead 8 (bass+lead)</td>
</tr>
<tr>
<td>electric grand</td>
<td>pizzicato strings</td>
<td>pad 1 (new age)</td>
</tr>
<tr>
<td>honky-tonk</td>
<td>orchestral harp</td>
<td>pad 2 (warm)</td>
</tr>
<tr>
<td>electric piano 1</td>
<td>timpani</td>
<td>pad 3 (polysynth)</td>
</tr>
<tr>
<td>electric piano 2</td>
<td>string ensemble 1</td>
<td>pad 4 (choir)</td>
</tr>
<tr>
<td>harpsichord</td>
<td>string ensemble 2</td>
<td>pad 5 (bowed)</td>
</tr>
<tr>
<td>clav</td>
<td>synthstrings 1</td>
<td>pad 6 (metallic)</td>
</tr>
<tr>
<td>celesta</td>
<td>synthstrings 2</td>
<td>pad 7 (halo)</td>
</tr>
<tr>
<td>glockenspiel</td>
<td>choir aahs</td>
<td>pad 8 (sweep)</td>
</tr>
<tr>
<td>music box</td>
<td>voice oohs</td>
<td>fx 1 (rain)</td>
</tr>
<tr>
<td>vibraphone</td>
<td>synth voice</td>
<td>fx 2 (soundtrack)</td>
</tr>
<tr>
<td>marimba</td>
<td>orchestra hit</td>
<td>fx 3 (crystal)</td>
</tr>
<tr>
<td>xylophone</td>
<td>trumpet</td>
<td>fx 4 (atmosphere)</td>
</tr>
<tr>
<td>tubular bells</td>
<td>trombone</td>
<td>fx 5 (brightness)</td>
</tr>
<tr>
<td>dulcimer</td>
<td>tuba</td>
<td>fx 6 (goblins)</td>
</tr>
<tr>
<td>drawbar organ</td>
<td>muted trumpet</td>
<td>fx 7 (echoes)</td>
</tr>
<tr>
<td>percussive organ</td>
<td>french horn</td>
<td>fx 8 (sci-fi)</td>
</tr>
<tr>
<td>rock organ</td>
<td>brass section</td>
<td>sitar</td>
</tr>
<tr>
<td>church organ</td>
<td>synthbrass 1</td>
<td>banjo</td>
</tr>
<tr>
<td>reed organ</td>
<td>synthbrass 2</td>
<td>shamisen</td>
</tr>
<tr>
<td>accordion</td>
<td>soprano sax</td>
<td>koto</td>
</tr>
<tr>
<td>harmonica</td>
<td>alto sax</td>
<td>kalimba</td>
</tr>
<tr>
<td>concertina</td>
<td>tenor sax</td>
<td>bagpipe</td>
</tr>
<tr>
<td>acoustic guitar (nylon)</td>
<td>baritone sax</td>
<td>fiddle</td>
</tr>
<tr>
<td>acoustic guitar (steel)</td>
<td>oboe</td>
<td>shanai</td>
</tr>
<tr>
<td>electric guitar (jazz)</td>
<td>english horn</td>
<td>tinkle bell</td>
</tr>
<tr>
<td>electric guitar (clean)</td>
<td>bassoon</td>
<td>agogo</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th>Electric guitar (muted)</th>
<th>Clarinet</th>
<th>Steel drums</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overdriven guitar</td>
<td>Piccolo</td>
<td>Woodblock</td>
</tr>
<tr>
<td>Distorted guitar</td>
<td>Flute</td>
<td>Taiko drum</td>
</tr>
<tr>
<td>Guitar harmonics</td>
<td>Recorder</td>
<td>Melodic tom</td>
</tr>
<tr>
<td>Acoustic bass</td>
<td>Pan flute</td>
<td>Synth drum</td>
</tr>
<tr>
<td>Electric bass (finger)</td>
<td>Blown bottle</td>
<td>Reverse cymbal</td>
</tr>
<tr>
<td>Electric bass (pick)</td>
<td>Shakuhachi</td>
<td>Guitar fret noise</td>
</tr>
<tr>
<td>Fretless bass</td>
<td>Whistle</td>
<td>Breath noise</td>
</tr>
<tr>
<td>Slap bass 1</td>
<td>Ocarina</td>
<td>Seashore</td>
</tr>
<tr>
<td>Slap bass 2</td>
<td>Lead 1 (square)</td>
<td>Bird tweet</td>
</tr>
<tr>
<td>Synth bass 1</td>
<td>Lead 2 (sawtooth)</td>
<td>Telephone ring</td>
</tr>
<tr>
<td>Synth bass 2</td>
<td>Lead 3 (calliope)</td>
<td>Helicopter</td>
</tr>
<tr>
<td>Violin</td>
<td>Lead 4 (chiff)</td>
<td>Applause</td>
</tr>
<tr>
<td>Viola</td>
<td>Lead 5 (charang)</td>
<td>Gunshot</td>
</tr>
<tr>
<td>Cello</td>
<td>Lead 6 (voice)</td>
<td></td>
</tr>
</tbody>
</table>

A.7 List of colors

Normal colors

Usage syntax is detailed in [Coloring objects], page 232.

<table>
<thead>
<tr>
<th>Black</th>
<th>White</th>
<th>Red</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Cyan</td>
<td>Magenta</td>
<td>Yellow</td>
</tr>
<tr>
<td>Grey</td>
<td>Darkred</td>
<td>Darkgreen</td>
<td>Darkblue</td>
</tr>
<tr>
<td>Darkcyan</td>
<td>Darkmagenta</td>
<td>Darkyellow</td>
<td></td>
</tr>
</tbody>
</table>

CSS color names

CSS color names may be used as-is in string arguments.

<table>
<thead>
<tr>
<th>Aliceblue</th>
<th>Darkturquoise</th>
<th>Lightsalmon</th>
<th>Papayawhip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antiquewhite</td>
<td>Darkviolet</td>
<td>Lightseagreen</td>
<td>Peachpuff</td>
</tr>
<tr>
<td>Aqua</td>
<td>Deepink</td>
<td>Lightskyblue</td>
<td>Peru</td>
</tr>
<tr>
<td>Aquamarine</td>
<td>Deepskyblue</td>
<td>Lightslategrey</td>
<td>Pink</td>
</tr>
<tr>
<td>Azure</td>
<td>Dimgrey</td>
<td>Lightslategrey</td>
<td>Plum</td>
</tr>
<tr>
<td>Beige</td>
<td>Dimgrey</td>
<td>Lightsteelblue</td>
<td>Powderblue</td>
</tr>
<tr>
<td>Bisque</td>
<td>Dodgerblue</td>
<td>Lightyellow</td>
<td>Purple</td>
</tr>
<tr>
<td>Black</td>
<td>Firebrick</td>
<td>Lime</td>
<td>Rebeccapurple</td>
</tr>
<tr>
<td>Blanchedalmond</td>
<td>Floralwhite</td>
<td>Limegreen</td>
<td>Red</td>
</tr>
<tr>
<td>Blue</td>
<td>Forestgreen</td>
<td>Linen</td>
<td>Rosybrown</td>
</tr>
<tr>
<td>Blueviolet</td>
<td>Fuchsia</td>
<td>Magenta</td>
<td>Royalblue</td>
</tr>
<tr>
<td>Brown</td>
<td>Gainsboro</td>
<td>Maroon</td>
<td>Saddlebrown</td>
</tr>
<tr>
<td>Burlywood</td>
<td>Ghostwhite</td>
<td>Mediumaquamarine</td>
<td>Salmon</td>
</tr>
<tr>
<td>Cadetblue</td>
<td>Gold</td>
<td>Mediumblue</td>
<td>Sandybrown</td>
</tr>
<tr>
<td>Chartreuse</td>
<td>Goldenrod</td>
<td>Mediumorchid</td>
<td>Seagreen</td>
</tr>
<tr>
<td>Chocolate</td>
<td>Gray</td>
<td>Mediumpurple</td>
<td>Seashell</td>
</tr>
<tr>
<td>Coral</td>
<td>Green</td>
<td>Mediumseagreen</td>
<td>Sienna</td>
</tr>
<tr>
<td>Cornflowerblue</td>
<td>Greenyellow</td>
<td>Mediumslateblue</td>
<td>Silver</td>
</tr>
<tr>
<td>Cornsilk</td>
<td>Grey</td>
<td>Mediumspringgreen</td>
<td>Skyblue</td>
</tr>
<tr>
<td>Crimson</td>
<td>Honeysuckle</td>
<td>Mediumturquoise</td>
<td>Slateblue</td>
</tr>
<tr>
<td>Cyan</td>
<td>Hotpink</td>
<td>Mediumvioletred</td>
<td>Slategray</td>
</tr>
<tr>
<td>Darkblue</td>
<td>Indianred</td>
<td>Midnightblue</td>
<td>Slategrey</td>
</tr>
<tr>
<td>Darkcyan</td>
<td>Indigo</td>
<td>Mintcream</td>
<td>Snow</td>
</tr>
<tr>
<td>Darkgoldenrod</td>
<td>Ivory</td>
<td>Mistyrose</td>
<td>Springgreen</td>
</tr>
</tbody>
</table>
### CSS color definitions differ from X color names for the following colors: green, grey, maroon, purple.

**X color names**

X11 color names ([https://en.wikipedia.org/wiki/X11_color_names](https://en.wikipedia.org/wiki/X11_color_names)) offer a wider choice than CSS names. They come in several variants:

- Any name that is spelled as a single word with capitalization (e.g., ‘LightSlateBlue’) can also be spelled as space-separated words with or without capitalization (e.g., ‘light slate blue’).
- The word ‘grey’ can always be spelled ‘gray’ (e.g., ‘DarkSlateGray’), without any difference in the output.
- Some names can take a numerical suffix (e.g., ‘LightSalmon4’).

The following tables present all color names that may be used without a numerical suffix, and then the subset of these that may be used with such a suffix.

#### Color Names without a numerical suffix

<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AliceBlue</td>
<td>LawnGreen</td>
<td>OrangeRed</td>
<td>firebrick</td>
</tr>
<tr>
<td>AntiqueWhite</td>
<td>LemonChiffon</td>
<td>PaleGoldenrod</td>
<td>gainsboro</td>
</tr>
<tr>
<td>BlanchedAlmond</td>
<td>LightBlue</td>
<td>PaleGreen</td>
<td>gold</td>
</tr>
<tr>
<td>BlueViolet</td>
<td>LightCoral</td>
<td>PaleTurquoise</td>
<td>goldenrod</td>
</tr>
<tr>
<td>CadetBlue</td>
<td>LightCyan</td>
<td>PaleVioletRed</td>
<td>green</td>
</tr>
<tr>
<td>CornflowerBlue</td>
<td>LightGoldenrod</td>
<td>PapayaWhip</td>
<td>grey</td>
</tr>
<tr>
<td>DarkBlue</td>
<td>LightGoldenrodYellow</td>
<td>PeachPuff</td>
<td>honeydew</td>
</tr>
<tr>
<td>DarkCyan</td>
<td>LightGreen</td>
<td>PowderBlue</td>
<td>ivory</td>
</tr>
<tr>
<td>DarkGoldenrod</td>
<td>LightGrey</td>
<td>RosyBrown</td>
<td>khaki</td>
</tr>
<tr>
<td>DarkGreen</td>
<td>LightPink</td>
<td>RoyalBlue</td>
<td>lavender</td>
</tr>
<tr>
<td>DarkGrey</td>
<td>LightSalmon</td>
<td>SaddleBrown</td>
<td>linen</td>
</tr>
<tr>
<td>DarkKhaki</td>
<td>LightSeaGreen</td>
<td>SandyBrown</td>
<td>magenta</td>
</tr>
<tr>
<td>DarkMagenta</td>
<td>LightSkyBlue</td>
<td>SeaGreen</td>
<td>maroon</td>
</tr>
<tr>
<td>DarkOliveGreen</td>
<td>LightSlateBlue</td>
<td>SkyBlue</td>
<td>moccasin</td>
</tr>
<tr>
<td>DarkOrange</td>
<td>LightSlateGrey</td>
<td>SlateBlue</td>
<td>navy</td>
</tr>
<tr>
<td>DarkOrchid</td>
<td>LightSteelBlue</td>
<td>SlateGrey</td>
<td>orange</td>
</tr>
<tr>
<td>DarkRed</td>
<td>LightYellow</td>
<td>SpringGreen</td>
<td>orchid</td>
</tr>
<tr>
<td>DarkSalmon</td>
<td>LimeGreen</td>
<td>SteelBlue</td>
<td>peru</td>
</tr>
<tr>
<td>DarkSeaGreen</td>
<td>MediumAquamarine</td>
<td>VioletRed</td>
<td>pink</td>
</tr>
<tr>
<td>DarkSlateBlue</td>
<td>MediumBlue</td>
<td>WhiteSmoke</td>
<td>plum</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables 683

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Hex Value</th>
<th>RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DarkSlateGrey</td>
<td>MediumOrchid</td>
<td>YellowGreen</td>
</tr>
<tr>
<td>DarkTurquoise</td>
<td>MediumPurple</td>
<td>aquamarine</td>
</tr>
<tr>
<td>DarkViolet</td>
<td>MediumSeaGreen</td>
<td>azure</td>
</tr>
<tr>
<td>DeepPink</td>
<td>MediumSlateBlue</td>
<td>beige</td>
</tr>
<tr>
<td>DeepSkyBlue</td>
<td>MediumSpringGreen</td>
<td>bisque</td>
</tr>
<tr>
<td>DimGrey</td>
<td>MediumTurquoise</td>
<td>black</td>
</tr>
<tr>
<td>DodgerBlue</td>
<td>MediumVioletRed</td>
<td>blue</td>
</tr>
<tr>
<td>FloralWhite</td>
<td>MidnightBlue</td>
<td>brown</td>
</tr>
<tr>
<td>ForestGreen</td>
<td>MintCream</td>
<td>burlywood</td>
</tr>
<tr>
<td>GhostWhite</td>
<td>MistyRose</td>
<td>chartreuse</td>
</tr>
<tr>
<td>GreenYellow</td>
<td>NavajoWhite</td>
<td>chocolate</td>
</tr>
<tr>
<td>HotPink</td>
<td>NavyBlue</td>
<td>coral</td>
</tr>
<tr>
<td>IndianRed</td>
<td>OldLace</td>
<td>cornsilk</td>
</tr>
<tr>
<td>LavenderBlush</td>
<td>OliveDrab</td>
<td>cyan</td>
</tr>
</tbody>
</table>

### Color names with a numerical suffix

In the following names the suffix N must be an integer between 1 and 4, from lighter to darker shades:

- **AntiqueWhiteN**
- **LightSkyBlueN**
- **SteelBlueN**
- **KhakiN**
- **CadetBlueN**
- **LightSteelBlueN**
- **VioletRedN**
- **MagentaN**
- **DarkGoldenrodN**
- **LightYellowN**
- **AquamarineN**
- **MaroonN**
- **DarkOliveGreenN**
- **MediumOrchidN**
- **AzureN**
- **OrangeN**
- **DarkOrangeN**
- **MediumPurpleN**
- **BisqueN**
- **OrchidN**
- **DarkOrchidN**
- **MistyRoseN**
- **BlueN**
- **PinkN**
- **DarkSeaGreenN**
- **NavajoWhiteN**
- **BrownN**
- **PlumN**
- **DeepPinkN**
- **OliveDrabN**
- **BurlywoodN**
- **PurpleN**
- **DeepSkyBlueN**
- **OrangeRedN**
- **ChartreuseN**
- **RedN**
- **DodgerBlueN**
- **PaleGreenN**
- **ChocolateN**
- **SalmonN**
- **HotPinkN**
- **PaleTurquoiseN**
- **CoralN**
- **SeashellN**
- **IndianRedN**
- **PaleVioletRedN**
- **CornsilkN**
- **SiennaN**
- **LavenderBlushN**
- **PeachPuffN**
- **CyanN**
- **SnowN**
- **LemonChiffonN**
- **RosyBrownN**
- **FirebrickN**
- **TanN**
- **LightBlueN**
- **RoyalBlueN**
- **GoldN**
- **ThistleN**
- **LightCyanN**
- **SeaGreenN**
- **GoldenrodN**
- **TomatoN**
- **LightGoldenrodN**
- **SkyBlueN**
- **GreenN**
- **TurquoiseN**
- **LightPinkN**
- ** SlateBlueN**
- **HoneydewN**
- **WheatN**
- **LightSalmonN**
- **SpringGreenN**
- **IvoryN**
- **YellowN**

### Grey Scale

A grey scale can be obtained using:

```
greyN
```

Where N is in the range 0-100.

### A.8 The Emmentaler font

The Emmentaler font consists of two sub-sets of glyphs. “Feta”, used for classical notation and “Parmesan”, used for Ancient notation.

Any glyph within the Emmentaler font can be accessed directly by using text markup along with the name of the glyph (as shown in the tables above). For example:

```
g\markup {\musicglyph "scripts.segno" }
```
or
\markup {\musicglyph "five"}

For more information see Section 1.8.2 [Formatting text], page 252.

Clef glyphs

clefs.C

\clef \clef_change

clefs.varC

\clef \clef_change

clefs.F

\clef \clef_change

clefs.G

\clef \clef_change

clefs.GG

\clef \clef_change

clefs.tenorG

\clef \clef_change

clefs.percussion

\clef \clef_change

clefs.varpercussion

\clef \clef_change

clefs.tab

\clef \clef_change

Time Signature glyphs

timesig.C44

timesig.C22
### Number glyphs

<table>
<thead>
<tr>
<th>plus</th>
<th>+</th>
<th>comma</th>
<th>,</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyphen</td>
<td>-</td>
<td>period</td>
<td>.</td>
</tr>
<tr>
<td>zero</td>
<td>0</td>
<td>one</td>
<td>1</td>
</tr>
<tr>
<td>two</td>
<td>2</td>
<td>three</td>
<td>3</td>
</tr>
<tr>
<td>four</td>
<td>4</td>
<td>five</td>
<td>5</td>
</tr>
<tr>
<td>six</td>
<td>6</td>
<td>seven</td>
<td>7</td>
</tr>
<tr>
<td>eight</td>
<td>8</td>
<td>nine</td>
<td>9</td>
</tr>
</tbody>
</table>

### Accidental glyphs

<table>
<thead>
<tr>
<th>accidental.sharp</th>
<th>accidental .sharp.arrowup</th>
<th>accidental.sharp .sharp.arrowup</th>
</tr>
</thead>
<tbody>
<tr>
<td>accidental</td>
<td>accidental .sharp.arrowdown</td>
<td>accidental .sharp.arrowboth</td>
</tr>
<tr>
<td>accidental.sharp</td>
<td>accidental .slashslash.stem</td>
<td>accidental.sharp .slashslash.stem</td>
</tr>
<tr>
<td>accidental.sharp</td>
<td>accidental .slashslashslash.stem</td>
<td>accidental.sharp .slashslashslash.stem</td>
</tr>
<tr>
<td>accidental.sharp</td>
<td>accidental .slashslashslash.stem</td>
<td>accidental.sharp .slashslashslash.stem</td>
</tr>
<tr>
<td>accidental.sharp</td>
<td>accidental .slashslashslash.stem</td>
<td>accidental.sharp .slashslashslash.stem</td>
</tr>
<tr>
<td>accidental.doublesharpp</td>
<td>accidental.natural</td>
<td>accidental.natural</td>
</tr>
<tr>
<td>accidental</td>
<td>accidental .natural.arrowup</td>
<td>accidental .natural.arrowdown</td>
</tr>
<tr>
<td>accidental</td>
<td>accidental .natural.arrowboth</td>
<td>accidental.flat</td>
</tr>
<tr>
<td>accidentals.flat.arrowup</td>
<td>accidentals.flat.arrowdown</td>
<td></td>
</tr>
<tr>
<td>accidentals.flat.arrowboth</td>
<td>accidentals.flat.slash</td>
<td></td>
</tr>
<tr>
<td>accidentals.flat.slashslash</td>
<td>accidentals.mirroredflat.flat</td>
<td></td>
</tr>
<tr>
<td>accidentals.mirroredflat.backslash</td>
<td>accidentals.flatflat.slash</td>
<td></td>
</tr>
<tr>
<td>accidentals.rightparen</td>
<td>accidentals.leftparen</td>
<td></td>
</tr>
</tbody>
</table>

**Default Notehead glyphs**

| noteheads.uM2 | noteheads.dM2 |
| noteheads.sM1 | noteheads.s0 |
| noteheads.s1 | noteheads.s2 |

**Special Notehead glyphs**

| noteheads.sM1double | noteheads.s0diamond |
| noteheads.s1diamond | noteheads.s2diamond |
| noteheads.s0triangle | noteheads.d1triangle |
| noteheads.ultriangle | noteheads.u2triangle |
| noteheads.d2triangle | noteheads.s0slash |
### Shape-note Notehead glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>noteheads.s0do</code></td>
<td>Notehead for C</td>
</tr>
<tr>
<td><code>noteheads.u1do</code></td>
<td>Notehead for C sharp/D flat</td>
</tr>
<tr>
<td><code>noteheads.d1do</code></td>
<td>Notehead for D</td>
</tr>
<tr>
<td><code>noteheads.u2do</code></td>
<td>Notehead for D sharp/E flat</td>
</tr>
<tr>
<td><code>noteheads.d2do</code></td>
<td>Notehead for E</td>
</tr>
<tr>
<td><code>noteheads.s0doThin</code></td>
<td>Thin version of C</td>
</tr>
<tr>
<td><code>noteheads.u1doThin</code></td>
<td>Thin version of C sharp/D flat</td>
</tr>
<tr>
<td><code>noteheads.d1doThin</code></td>
<td>Thin version of D</td>
</tr>
<tr>
<td><code>noteheads.u2doThin</code></td>
<td>Thin version of D sharp/E flat</td>
</tr>
<tr>
<td><code>noteheads.d2doThin</code></td>
<td>Thin version of E</td>
</tr>
<tr>
<td><code>noteheads.s0re</code></td>
<td>Notehead for F</td>
</tr>
<tr>
<td><code>noteheads.u1re</code></td>
<td>Notehead for F sharp/G flat</td>
</tr>
<tr>
<td><code>noteheads.d1re</code></td>
<td>Notehead for G</td>
</tr>
<tr>
<td><code>noteheads.u2re</code></td>
<td>Notehead for G sharp/A flat</td>
</tr>
<tr>
<td><code>noteheads.d2re</code></td>
<td>Notehead for A</td>
</tr>
<tr>
<td><code>noteheads.s0reThin</code></td>
<td>Thin version of F</td>
</tr>
<tr>
<td><code>noteheads.u1reThin</code></td>
<td>Thin version of F sharp/G flat</td>
</tr>
<tr>
<td><code>noteheads.d1reThin</code></td>
<td>Thin version of G</td>
</tr>
<tr>
<td><code>noteheads.u2reThin</code></td>
<td>Thin version of G sharp/A flat</td>
</tr>
<tr>
<td><code>noteheads.d2reThin</code></td>
<td>Thin version of A</td>
</tr>
<tr>
<td><code>noteheads.s0mi</code></td>
<td>Notehead for B</td>
</tr>
<tr>
<td><code>noteheads.s1mi</code></td>
<td>Notehead for C</td>
</tr>
<tr>
<td>noteheads.s2mi</td>
<td>noteheads.s0miMirror</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>noteheads.s1miMirror</td>
<td>noteheads.s2miMirror</td>
</tr>
<tr>
<td>noteheads.s0miThin</td>
<td>noteheads.s1miThin</td>
</tr>
<tr>
<td>noteheads.s2miThin</td>
<td>noteheads.u0fa</td>
</tr>
<tr>
<td>noteheads.d0fa</td>
<td>noteheads.u1fa</td>
</tr>
<tr>
<td>noteheads.d1fa</td>
<td>noteheads.u2fa</td>
</tr>
<tr>
<td>noteheads.d2fa</td>
<td>noteheads.u0faThin</td>
</tr>
<tr>
<td>noteheads.d0faThin</td>
<td>noteheads.u1faThin</td>
</tr>
<tr>
<td>noteheads.d1faThin</td>
<td>noteheads.u2faThin</td>
</tr>
<tr>
<td>noteheads.d2faThin</td>
<td>noteheads.s0sol</td>
</tr>
<tr>
<td>noteheads.s1sol</td>
<td>noteheads.s2sol</td>
</tr>
<tr>
<td>noteheads.s0la</td>
<td>noteheads.s1la</td>
</tr>
<tr>
<td>noteheads.s2la</td>
<td>noteheads.s0laThin</td>
</tr>
<tr>
<td>noteheads.s1laThin</td>
<td>noteheads.s2laThin</td>
</tr>
<tr>
<td>noteheads.s0ti</td>
<td>noteheads.ulti</td>
</tr>
</tbody>
</table>
noteheads.d1ti ◆ noteheads.u2ti ◆

noteheads.d2ti ◆ noteheads.s0tiThin ◆

noteheads.ultiThin ◆ noteheads.d1tiThin ◆

noteheads.u2tiThin ◆ noteheads.d2tiThin ◆

noteheads.u0doFunk ◆ noteheads.d0doFunk ◆

noteheads.u1doFunk ◆ noteheads.d1doFunk ◆

noteheads.u2doFunk ◆ noteheads.d2doFunk ◆

noteheads.u0reFunk ◆ noteheads.d0reFunk ◆

noteheads.u1reFunk ◆ noteheads.d1reFunk ◆

noteheads.u2reFunk ◆ noteheads.d2reFunk ◆

noteheads.u0miFunk ◆ noteheads.d0miFunk ◆

noteheads.u1miFunk ◆ noteheads.d1miFunk ◆

noteheads.s2miFunk ◆ noteheads.u0faFunk ◆

noteheads.d0faFunk ◆ noteheads.ulfaFunk ◆

noteheads.d1faFunk ◆ noteheads.u2faFunk ◆
Appendix A: Notation manual tables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.d2faFunk</td>
<td>noteheads.s0solFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s1solFunk</td>
<td>noteheads.s2solFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0laFunk</td>
<td>noteheads.s1laFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s2laFunk</td>
<td>noteheads.u0tiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d0tiFunk</td>
<td>noteheads.ultiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1tiFunk</td>
<td>noteheads.u2tiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2tiFunk</td>
<td>noteheads.s0doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1doWalker</td>
<td>noteheads.d1doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2doWalker</td>
<td>noteheads.d2doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0reWalker</td>
<td>noteheads.u1reWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1reWalker</td>
<td>noteheads.u2reWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2reWalker</td>
<td>noteheads.s0miWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s1miWalker</td>
<td>noteheads.s2miWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0faWalker</td>
<td>noteheads.ulfaWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1faWalker</td>
<td>noteheads.u2faWalker</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

Rest glyphs

<table>
<thead>
<tr>
<th>Noteheads</th>
<th>Rests</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.d2faWalker</td>
<td>noteheads.s0laWalker</td>
</tr>
<tr>
<td>noteheads.s1laWalker</td>
<td>noteheads.s2laWalker</td>
</tr>
<tr>
<td>noteheads.s0tiWalker</td>
<td>noteheads.ultiWalker</td>
</tr>
<tr>
<td>noteheads.d1tiWalker</td>
<td>noteheads.u2tiWalker</td>
</tr>
<tr>
<td>noteheads.d2tiWalker</td>
<td></td>
</tr>
<tr>
<td>rests.0</td>
<td>rests.1</td>
</tr>
<tr>
<td>rests.0o</td>
<td>rests.1o</td>
</tr>
<tr>
<td>rests.M3</td>
<td>rests.M2</td>
</tr>
<tr>
<td>rests.M1</td>
<td>rests.M1o</td>
</tr>
<tr>
<td>rests.2</td>
<td>rests.2classical</td>
</tr>
<tr>
<td>rests.2z</td>
<td>rests.3</td>
</tr>
<tr>
<td>rests.4</td>
<td>rests.5</td>
</tr>
<tr>
<td>rests.6</td>
<td>rests.7</td>
</tr>
<tr>
<td>rests.8</td>
<td>rests.9</td>
</tr>
<tr>
<td>rests.10</td>
<td></td>
</tr>
</tbody>
</table>
Flag glyphs

flags.u3 / flags.u4
flags.u5 / flags.u6
flags.u7 / flags.u8
flags.u9 / flags.u10
flags.d3 / flags.d4
flags.d5 / flags.d6
flags.d7 / flags.d8
flags.d9 / flags.d10
flags.ugrace / flags.dgrace

Dot glyphs

dots.dot
Dynamic glyphs

<table>
<thead>
<tr>
<th>Space</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>p</td>
<td>p</td>
<td>r</td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>z</td>
</tr>
</tbody>
</table>

Script glyphs

<table>
<thead>
<tr>
<th>scripts.ufermata</th>
<th>scripts.dfermata</th>
</tr>
</thead>
<tbody>
<tr>
<td>scripts.uhenzeshortfermata</td>
<td>scripts.dhenzeshortfermata</td>
</tr>
<tr>
<td>scripts.uhenzelongfermata</td>
<td>scripts.dhenzelongfermata</td>
</tr>
<tr>
<td>scripts.ushortfermata</td>
<td>scripts.dshortfermata</td>
</tr>
<tr>
<td>scripts.uveryshortfermata</td>
<td>scripts.dveryshortfermata</td>
</tr>
<tr>
<td>scripts.ulongfermata</td>
<td>scripts.dlongfermata</td>
</tr>
<tr>
<td>scripts.uverylongfermata</td>
<td>scripts.dverylongfermata</td>
</tr>
<tr>
<td>scripts.thumb</td>
<td>scripts.sforzato</td>
</tr>
<tr>
<td>scripts.espr</td>
<td>scripts.staccato</td>
</tr>
<tr>
<td>scripts.ustaccatissimo</td>
<td>scripts.dstaccatissimo</td>
</tr>
<tr>
<td>Script</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td><code>scripts.tenuto</code></td>
<td>tenuto slash</td>
</tr>
<tr>
<td><code>scripts.dportato</code></td>
<td>dportato semicircle</td>
</tr>
<tr>
<td><code>scripts.dmarcato</code></td>
<td>dmarcato semicircle</td>
</tr>
<tr>
<td><code>scripts.upedaltoe</code></td>
<td>upedaltoe semicircle</td>
</tr>
<tr>
<td><code>scripts.upedalheel</code></td>
<td>upedalheel horizontal semi-circle</td>
</tr>
<tr>
<td><code>scripts.haydnturn</code></td>
<td>haydn turn</td>
</tr>
<tr>
<td><code>scripts.flageolet</code></td>
<td>flageolet</td>
</tr>
<tr>
<td><code>scripts.turn</code></td>
<td>turn</td>
</tr>
<tr>
<td><code>scripts.halfopenvertical</code></td>
<td>halfopen vertical</td>
</tr>
<tr>
<td><code>scripts.stopped</code></td>
<td>stopped</td>
</tr>
<tr>
<td><code>scripts.downbow</code></td>
<td>downbow</td>
</tr>
<tr>
<td><code>scripts.upbow</code></td>
<td>upbow</td>
</tr>
<tr>
<td><code>scripts.reverseturn</code></td>
<td>reverseturn</td>
</tr>
<tr>
<td><code>scripts.untilturn</code></td>
<td>untilturn</td>
</tr>
<tr>
<td><code>scripts.turn</code></td>
<td>turn</td>
</tr>
<tr>
<td><code>scripts.slashturn</code></td>
<td>slashturn</td>
</tr>
<tr>
<td><code>scripts.haydnturn</code></td>
<td>haydn turn</td>
</tr>
<tr>
<td><code>scripts.trill</code></td>
<td>trill</td>
</tr>
<tr>
<td><code>scripts.flageolet</code></td>
<td>flageolet</td>
</tr>
<tr>
<td><code>scripts.segno</code></td>
<td>segno</td>
</tr>
<tr>
<td><code>scripts.varsegno</code></td>
<td>varsegno</td>
</tr>
<tr>
<td><code>scripts.coda</code></td>
<td>coda</td>
</tr>
<tr>
<td><code>scripts.varcoda</code></td>
<td>varcoda</td>
</tr>
<tr>
<td><code>scripts.rcomma</code></td>
<td>rcomma</td>
</tr>
<tr>
<td><code>scripts.lcomma</code></td>
<td>lcomma</td>
</tr>
<tr>
<td><code>scripts.rvarcomma</code></td>
<td>rvarcomma</td>
</tr>
<tr>
<td><code>scripts.lvarcomma</code></td>
<td>lvarcomma</td>
</tr>
<tr>
<td><code>scripts.arpeggio</code></td>
<td>arpeggio</td>
</tr>
<tr>
<td><code>scripts.open</code></td>
<td>open</td>
</tr>
<tr>
<td><code>scripts.halfopen</code></td>
<td>halfopen</td>
</tr>
<tr>
<td><code>scripts.dportato</code></td>
<td>dportato</td>
</tr>
<tr>
<td><code>scripts.dmarcato</code></td>
<td>dmarcato</td>
</tr>
<tr>
<td><code>scripts.upedaltoe</code></td>
<td>upedaltoe</td>
</tr>
<tr>
<td><code>scripts.upedalheel</code></td>
<td>upedalheel</td>
</tr>
<tr>
<td><code>scripts.haydnturn</code></td>
<td>haydn turn</td>
</tr>
<tr>
<td><code>scripts.trill</code></td>
<td>trill</td>
</tr>
<tr>
<td><code>scripts.flageolet</code></td>
<td>flageolet</td>
</tr>
<tr>
<td><code>scripts.segno</code></td>
<td>segno</td>
</tr>
<tr>
<td><code>scripts.varsegno</code></td>
<td>varsegno</td>
</tr>
<tr>
<td><code>scripts.coda</code></td>
<td>coda</td>
</tr>
<tr>
<td><code>scripts.varcoda</code></td>
<td>varcoda</td>
</tr>
<tr>
<td><code>scripts.rcomma</code></td>
<td>rcomma</td>
</tr>
<tr>
<td><code>scripts.lcomma</code></td>
<td>lcomma</td>
</tr>
<tr>
<td><code>scripts.rvarcomma</code></td>
<td>rvarcomma</td>
</tr>
<tr>
<td><code>scripts.lvarcomma</code></td>
<td>lvarcomma</td>
</tr>
<tr>
<td><code>scripts.arpeggio</code></td>
<td>arpeggio</td>
</tr>
<tr>
<td>Script</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>scripts.trill_element</td>
<td>scripts.arpeggio</td>
</tr>
<tr>
<td>scripts.arpeggio.arrow.1</td>
<td>scripts.trillelement</td>
</tr>
<tr>
<td>scripts.prall</td>
<td>scripts.mordent</td>
</tr>
<tr>
<td>scripts.prallprall</td>
<td>scripts.prallmordent</td>
</tr>
<tr>
<td>scripts.upprall</td>
<td>scripts.upmordent</td>
</tr>
<tr>
<td>scripts.prallup</td>
<td>scripts.downprall</td>
</tr>
<tr>
<td>scripts.downmordent</td>
<td>scripts.pralldown</td>
</tr>
<tr>
<td>scripts.lineprall</td>
<td>scripts.caesura.curved</td>
</tr>
<tr>
<td>scripts.caesura.straight</td>
<td>scripts.tickmark</td>
</tr>
<tr>
<td>scripts.snappizzicato</td>
<td>scripts.ictus</td>
</tr>
<tr>
<td>scripts.uaccentus</td>
<td>scripts.daccentus</td>
</tr>
<tr>
<td>scripts.usemicirculus</td>
<td>scripts.dsemicirculus</td>
</tr>
<tr>
<td>scripts.circulus</td>
<td>scripts.augmentum</td>
</tr>
<tr>
<td>scripts.usignumcongruentiae</td>
<td>scripts.dsignumcongruentiae</td>
</tr>
</tbody>
</table>
### Arrowhead glyphs

<table>
<thead>
<tr>
<th>Arrowhead Glyphs</th>
<th>Arrowhead Glyphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arrowheads.open.01</code></td>
<td><code>arrowheads.open.0M1</code></td>
</tr>
<tr>
<td><code>arrowheads.open.11</code></td>
<td><code>arrowheads.open.1M1</code></td>
</tr>
<tr>
<td><code>arrowheads.close.01</code></td>
<td><code>arrowheads.close.0M1</code></td>
</tr>
<tr>
<td><code>arrowheads.close.11</code></td>
<td><code>arrowheads.close.1M1</code></td>
</tr>
</tbody>
</table>

### Bracket-tip glyphs

- `brackettips.up` ➤ `brackettips.down`

### Pedal glyphs

<table>
<thead>
<tr>
<th>Pedal Glyphs</th>
<th>Pedal Glyphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pedal.*</code></td>
<td><code>pedal.M</code></td>
</tr>
<tr>
<td><code>pedal..</code></td>
<td><code>pedal.P</code></td>
</tr>
<tr>
<td><code>pedal.d</code></td>
<td><code>pedal.e</code></td>
</tr>
<tr>
<td><code>pedal.Ped</code></td>
<td></td>
</tr>
</tbody>
</table>

### Accordion glyphs

<table>
<thead>
<tr>
<th>Accordion Glyphs</th>
<th>Accordion Glyphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>accordion.discant</code></td>
<td><code>accordion.dot</code></td>
</tr>
<tr>
<td><code>accordion.freebass</code></td>
<td><code>accordion.stdbass</code></td>
</tr>
<tr>
<td><code>accordion.bayanbass</code></td>
<td><code>accordion.oldEE</code></td>
</tr>
<tr>
<td><code>accordion.push</code></td>
<td><code>accordion.pull</code></td>
</tr>
</tbody>
</table>
### Tie glyphs

<table>
<thead>
<tr>
<th>ties.lyric.short</th>
<th>ties.lyric.default</th>
</tr>
</thead>
</table>

### Vaticana glyphs

<table>
<thead>
<tr>
<th>clefs.vaticana.do</th>
<th>clefs.vaticana.do_change</th>
</tr>
</thead>
<tbody>
<tr>
<td>clefs.vaticana.fa</td>
<td>clefs.vaticana.fa_change</td>
</tr>
<tr>
<td>custodes.vaticana.u0</td>
<td>custodes.vaticana.u1</td>
</tr>
<tr>
<td>custodes.vaticana.u2</td>
<td>custodes.vaticana.d0</td>
</tr>
<tr>
<td>custodes.vaticana.d1</td>
<td>custodes.vaticana.d2</td>
</tr>
<tr>
<td>accidentals.vaticanaM1</td>
<td>accidentals.vaticana0</td>
</tr>
<tr>
<td>dots.dotvaticana</td>
<td>noteheads.svaticana.punctum</td>
</tr>
<tr>
<td>noteheads.svaticana.punctum.cavum</td>
<td>noteheads.svaticana.linea.punctum</td>
</tr>
<tr>
<td>noteheads.svaticana.linea.punctum.cavum</td>
<td>noteheads.svaticana.inclinatum</td>
</tr>
<tr>
<td>noteheads.svaticana.ipes</td>
<td>noteheads.svaticana.vlpes</td>
</tr>
<tr>
<td>noteheads.svaticana.ipes</td>
<td>noteheads.svaticana.vpes</td>
</tr>
<tr>
<td>noteheads.svaticana.plica</td>
<td>noteheads.svaticana.vplica</td>
</tr>
<tr>
<td>noteheads.svaticana.epiphonus</td>
<td>noteheads.svaticana.vepiphonus</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

Medicaea glyphs

- clefs.medicaea.do
- clefs.medicaea.do_change
- clefs.medicaea.fa
- clefs.medicaea.fa_change
- custodes.medicaea.u0
- custodes.medicaea.u1
- custodes.medicaea.u2
- custodes.medicaea.d0
- custodes.medicaea.d1
- custodes.medicaea.d2
- accidentals.medicaeaM1
- noteheads.smedicaea.inclinatum
- noteheads.smedicaea.punctum
- noteheads.smedicaea.rvirga
- noteheads.smedicaea.virga

Hufnagel glyphs

- clefs.hufnagel.do
- clefs.hufnagel.do_change
- clefs.hufnagel.fa
- clefs.hufnagel.fa_change
- clefs.hufnagel.do.fa
- clefs.hufnagel.do.fa_change
### Mensural Glyphs

- custodes.hufnagel.u0
- custodes.hufnagel.u2
- custodes.hufnagel.d1
- accidentals.hufnagelM1
- noteheads
- noteheads.shufnagel.punctum
- noteheads.shufnagel.virga
- noteheads.shufnagel.lpes
- rests.M3mensural
- rests.M2mensural
- rests.M1mensural
- rests.0mensural
- rests.1mensural
- rests.2mensural
- rests.3mensural
- rests.4mensural
- clefs.mensural.c
- clefs.mensural.c_change
- clefs.mensural.f
- clefs.mensural.f_change
- clefs.mensural.g
- clefs.mensural.g_change
- custodes.mensural.u0
- custodes.mensural.u1
- custodes.mensural.u2
- custodes.mensural.d0
- custodes.mensural.d1
- custodes.mensural.d2
- custodes.mensural.d3
- custodes.mensural.d4
<table>
<thead>
<tr>
<th>timesig.mensural44</th>
<th>C</th>
<th>timesig.mensural22</th>
<th>(\ominus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>timesig.mensural32</td>
<td>°</td>
<td>timesig.mensural64</td>
<td>C</td>
</tr>
<tr>
<td>timesig.mensural94</td>
<td>°</td>
<td>timesig.mensural34</td>
<td>(\phi)</td>
</tr>
<tr>
<td>timesig.mensural68</td>
<td>(\phi)</td>
<td>timesig.mensural98</td>
<td>(\phi)</td>
</tr>
<tr>
<td>timesig.mensural48</td>
<td>°</td>
<td>timesig.mensural68alt</td>
<td>°</td>
</tr>
<tr>
<td>timesig.mensural24</td>
<td>(\phi)</td>
<td>noteheads.uM3mensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.dM3mensural</td>
<td>(\Ll)</td>
<td>noteheads.sM3ligmensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.uM2mensural</td>
<td>(\Ll)</td>
<td>noteheads.dM2mensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.sM2ligmensural</td>
<td>(\Ll)</td>
<td>noteheads.sM1mensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.urM3mensural</td>
<td>(\Ll)</td>
<td>noteheads.drM3mensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.srM3ligmensural</td>
<td>(\Ll)</td>
<td>noteheads.urM2mensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.drM2mensural</td>
<td>(\Ll)</td>
<td>noteheads.srM2ligmensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.srM1mensural</td>
<td>(\Ll)</td>
<td>noteheads.urM3semimensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.dM3semimensural</td>
<td>(\Ll)</td>
<td>noteheads.sM3semiligmensural</td>
<td>(\Ll)</td>
</tr>
<tr>
<td>noteheads.uM2semimensural</td>
<td>(\Ll)</td>
<td>noteheads.sM3semimensural</td>
<td>(\Ll)</td>
</tr>
</tbody>
</table>
Neomensural glyphs

rests.M3neomensural || rests.M2neomensural |

rests.M1neomensural |

rests.1neomensural |

rests.3neomensural |

clefs.neomensural.c || clefs.neomensural.c_change |
<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timesig.neomensural44</td>
<td>C</td>
</tr>
<tr>
<td>timesig.neomensural32</td>
<td>O</td>
</tr>
<tr>
<td>timesig.neomensural94</td>
<td>O</td>
</tr>
<tr>
<td>timesig.neomensural68</td>
<td>C</td>
</tr>
<tr>
<td>timesig.neomensural48</td>
<td>C</td>
</tr>
<tr>
<td>noteheads.uM3neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.dM3neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.sM1neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.urM3neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.drM3neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.urM2neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.drM2neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.srM1neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.s0neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.s1neomensural</td>
<td>II</td>
</tr>
<tr>
<td>noteheads.s2neomensural</td>
<td>II</td>
</tr>
</tbody>
</table>

**Petrucci glyphs**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clefs.petrucci.c1</td>
<td>I</td>
</tr>
<tr>
<td>clefs.petrucci.c1_change</td>
<td>I</td>
</tr>
<tr>
<td>clefs.petrucci.c2</td>
<td>I</td>
</tr>
<tr>
<td>clefs.petrucci.c2_change</td>
<td>I</td>
</tr>
<tr>
<td>clefs.petrucci.c3</td>
<td>clefs.petrucci.c3_change</td>
</tr>
<tr>
<td>clefs.petrucci.c4</td>
<td>clefs.petrucci.c4_change</td>
</tr>
<tr>
<td>clefs.petrucci.c5</td>
<td>clefs.petrucci.c5_change</td>
</tr>
<tr>
<td>clefs.petrucci.f</td>
<td>clefs.petrucci.f_change</td>
</tr>
<tr>
<td>clefs.petrucci.g</td>
<td>clefs.petrucci.g_change</td>
</tr>
</tbody>
</table>

| noteheads.s0petrucci | noteheads.s1petrucci   |
| noteheads.s2petrucci | noteheads.s0blackpetrucci |

| noteheads.ssolesmes | noteheads.ssolesmes.auct.asc |
| noteheads.ssolesmes.incl.parvum | noteheads.ssolesmes.auct.asc |
| noteheads.ssolesmes.auct.desc | noteheads.ssolesmes.incl.auctum |
| noteheads.ssolesmes.stropha | noteheads.ssolesmes.stropha.aucta |
| noteheads.ssolesmes.oriscus | |

**Solesmes glyphs**
Kievan Notation glyphs

-clefs.kievan.do- clefs.kievan.do_change-
-accidentals.kievan1- accidentals.kievanM1-
-scripts.barline.kievan- dots.dotkievan-
-noteheads.sM2kievan- noteheads.sM1kievan-
-noteheads.s0kievan- noteheads.s1kievan-
-noteheads.u2kievan- noteheads.s1kievan-

A.9 Note head styles

The following styles may be used for note heads.

default altdefault

baroque neomensural

mensural petrucci
Appendix A: Notation manual tables

A.10 Clef styles

The following table shows all the clef styles possible (including where middle C sits relative to the clef).

### Standard clefs

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef G</td>
<td><img src="#" alt="clef G" /></td>
<td>\clef &quot;G2&quot;</td>
<td>![clef &quot;G2&quot;]</td>
</tr>
<tr>
<td>\clef treble</td>
<td><img src="#" alt="clef treble" /></td>
<td>\clef violin</td>
<td><img src="#" alt="clef violin" /></td>
</tr>
<tr>
<td>\clef french</td>
<td><img src="#" alt="clef french" /></td>
<td>\clef GG</td>
<td><img src="#" alt="clef GG" /></td>
</tr>
<tr>
<td>\clef tenorG</td>
<td><img src="#" alt="clef tenorG" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


## Appendix A: Notation manual tables

### Clefs

<table>
<thead>
<tr>
<th>Clef</th>
<th>Output</th>
<th>Clef</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef soprano</td>
<td><img src="image" alt="Soprano Clef" /></td>
<td>\clef mezzosoprano</td>
<td><img src="image" alt="Mezzo Soprano Clef" /></td>
</tr>
<tr>
<td>\clef C</td>
<td><img src="image" alt="C Clef" /></td>
<td>\clef alto</td>
<td><img src="image" alt="Alto Clef" /></td>
</tr>
<tr>
<td>\clef tenor</td>
<td><img src="image" alt="Tenor Clef" /></td>
<td>\clef baritone</td>
<td><img src="image" alt="Baritone Clef" /></td>
</tr>
<tr>
<td>\clef varC</td>
<td><img src="image" alt="VarC Clef" /></td>
<td>\clef altopvarC</td>
<td><img src="image" alt="VarC Alto Clef" /></td>
</tr>
<tr>
<td>\clef tenorvarC</td>
<td><img src="image" alt="Tenor VarC Clef" /></td>
<td>\clef baritonevarC</td>
<td><img src="image" alt="Baritone VarC Clef" /></td>
</tr>
<tr>
<td>\clef varbaritone</td>
<td><img src="image" alt="VarBaritone Clef" /></td>
<td>\clef baritonevarF</td>
<td><img src="image" alt="Baritone VarF Clef" /></td>
</tr>
<tr>
<td>\clef F</td>
<td><img src="image" alt="F Clef" /></td>
<td>\clef bass</td>
<td><img src="image" alt="Bass Clef" /></td>
</tr>
<tr>
<td>\clef subbass</td>
<td><img src="image" alt="Subbass Clef" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Percussion staff clef

### Tab staff clefs

Example | Output | Example | Output
--------|--------|---------|--------
\clef percussion | ![Percussion Clef](image) | \clef varpercussion | ![Varpercussion Clef](image) |
### Example Output Example Output
\new TabStaff {
  \clef tab
}\new TabStaff {
  \clef moderntab
}

### Ancient music clefs

**Gregorian**

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef &quot;vaticana-do1&quot;</td>
<td><img src="image1" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;vaticana-do2&quot;</td>
<td><img src="image2" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;vaticana-do3&quot;</td>
<td><img src="image3" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;vaticana-fa1&quot;</td>
<td><img src="image4" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;vaticana-fa2&quot;</td>
<td><img src="image5" alt="Example" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef &quot;medicaea-do1&quot;</td>
<td><img src="image6" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;medicaea-do2&quot;</td>
<td><img src="image7" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;medicaea-do3&quot;</td>
<td><img src="image8" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;medicaea-fa1&quot;</td>
<td><img src="image9" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;medicaea-fa2&quot;</td>
<td><img src="image10" alt="Example" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef &quot;hufnagel-do1&quot;</td>
<td><img src="image11" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;hufnagel-do2&quot;</td>
<td><img src="image12" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;hufnagel-do3&quot;</td>
<td><img src="image13" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;hufnagel-fa1&quot;</td>
<td><img src="image14" alt="Example" /></td>
</tr>
<tr>
<td>\clef &quot;hufnagel-fa2&quot;</td>
<td><img src="image15" alt="Example" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\clef &quot;hufnagel-do-fa&quot;</td>
<td><img src="image16" alt="Example" /></td>
</tr>
</tbody>
</table>
## Mensural

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ clef &quot;mensural-c1&quot;</td>
<td>\</td>
<td>\ clef &quot;mensural-c2&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;mensural-c3&quot;</td>
<td>\</td>
<td>\ clef &quot;mensural-c4&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;mensural-c5&quot;</td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;mensural-f&quot;</td>
<td>\</td>
<td>\ clef &quot;mensural-g&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;blackmensural-c1&quot;</td>
<td>\</td>
<td>\ clef &quot;blackmensural-c2&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;blackmensural-c3&quot;</td>
<td>\</td>
<td>\ clef &quot;blackmensural-c4&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;blackmensural-c5&quot;</td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;neomensural-c1&quot;</td>
<td>\</td>
<td>\ clef &quot;neomensural-c2&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td>\ clef &quot;neomensural-c3&quot;</td>
<td>\</td>
<td>\ clef &quot;neomensural-c4&quot;</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>\</td>
<td></td>
<td>\</td>
</tr>
</tbody>
</table>
\clef "neomensural-c5"
\clef "petrucci-c1" \clef "petrucci-c2"
\clef "petrucci-c3" \clef "petrucci-c4"
\clef "petrucci-c5"
\clef "petrucci-f" \clef "petrucci-f2"
\clef "petrucci-f3" \clef "petrucci-f4"
\clef "petrucci-f5"
\clef "petrucci-g1" \clef "petrucci-g2"
\clef "petrucci-g"

Kievan
Example: \clef "kievan-do"

\clef "kievan-do"

A.11 Text markup commands

The following commands can all be used inside \markup { }.

A.11.1 Font

\abs-fontsize size (number) arg (markup)

Use size as the absolute font size (in points) to display arg. Adjusts baseline-skip and word-space accordingly.

\markup {%
  default text font size
  hspace #2
  \abs-fontsize #16 \textfontsize16
  hspace #2
  \abs-fontsize #12 \textfontsize12
%
}

default text font size \textfontsize16 \textfontsize12

Used properties:
• baseline-skip (3)
• word-space (0.6)

\bold arg (markup)

Switch to bold font-series.

\markup {
  default
  hspace #2
  \bold
  bold
}

default \bold

\box arg (markup)

Draw a box round arg. Looks at thickness, box-padding and font-size properties to determine line thickness and padding around the markup.

\markup {
  \override #'(box-padding . 0.5)
  box
  \line { V. S. }
}

V. S.

Used properties:
• box-padding (0.2)
Appendix A: Notation manual tables

- font-size (0)
- thickness (1)

\caps arg (markup)
Copy of the \smallCaps command.

\markup {
  \default
  \hspace #2
  \caps {
    Text in small caps
  }
}

\default  TEXT IN SMALL CAPS

\dynamic arg (markup)
Use the dynamic font. This font only contains s, f, m, z, p, and r. When producing phrases, like ‘più f’, the normal words (like ‘più’) should be done in a different font. The recommended font for this is bold and italic.

\markup {
  \default
  \dynamic {
    \sfzp
  }
}

\sfzp

\finger arg (markup)
Set arg as small numbers.

\markup {
  \default
  \finger {
    1 2 3 4 5
  }
}

1 2 3 4 5

\fontCaps arg (markup)
Set font-shape to caps
Note: \fontCaps requires the installation and selection of fonts which support the caps font shape.

\fontsize increment (number) arg (markup)
Add increment to the font-size. Adjusts baseline-skip accordingly.

\markup {
  \default
  \hspace #2
  \fontsize #-1.5 smaller
}

\default  smaller
Used properties:

- baseline-skip (2)
- word-space (1)
- font-size (0)

\hugearg (markup)
Set font size to +2.
\markup {
  \default
  \hspace #2
  \huge
  \huge
}

\default huge

\italicarg (markup)
Use italic font-shape for arg.
\markup {
  \default
  \hspace #2
  \italic
  \italic
}

\default italic

\largearg (markup)
Set font size to +1.
\markup {
  \default
  \hspace #2
  \large
  \large
}

\default large

\largerarg (markup)
Increase the font size relative to the current setting.
\markup {
  \default
  \hspace #2
  \larger
  \larger
}

\default larger

\magnify sz (number) arg (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:
\[ A \magnify #1.1 \{ A \} A \]
Note: Magnification only works if a font name is explicitly selected. Use \texttt{\textbackslash fontsize} otherwise.

\begin{verbatim}
\markup {
  default
  \hspace #2
  \magnify #1.5 {  
    50\% larger
  }
}
\end{verbatim}

\texttt{default 50\% larger}

\texttt{\textbackslash medium arg (markup)}

Switch to medium font-series (in contrast to bold).

\begin{verbatim}
\markup {
  \bold {
    some bold text
    \hspace #2
    \texttt{\textbackslash medium {  
      medium font series
    }}
    \hspace #2
    bold again
  }
}
\end{verbatim}

\texttt{some bold text medium font series bold again}

\texttt{\textbackslash normal-size-sub arg (markup)}

Set \texttt{arg} in subscript with a normal font size.

\begin{verbatim}
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
\end{verbatim}

\texttt{default subscript in standard size}

Used properties:

- font-size (0)

\texttt{\textbackslash normal-size-super arg (markup)}

Set \texttt{arg} in superscript with a normal font size.

\begin{verbatim}
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
\end{verbatim}

\texttt{default superscript in standard size}
Used properties:
- \font-size(0)

\normal-text arg (markup)
Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

\begin{verbatim}
\markup {
  \huge \bold \sans \caps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
    \hspace #2
    as before
  }
}
\end{verbatim}

\textbf{HUGE BOLD SANS CAPS} \, \textit{huge normal} \, \textit{AS BEFORE}

\normalsize arg (markup)
Set font size to default.

\begin{verbatim}
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
\end{verbatim}

\textit{this is very small} \, \textbf{normal size} \, \textit{teeny again}

\number arg (markup)
Set font family to number, which yields the font used for time signatures and fingerings. This font contains numbers and some punctuation; it has no letters.

\begin{verbatim}
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
\end{verbatim}

\textbf{0123456789.},

\overtie arg (markup)
Overtie arg.

\begin{verbatim}
\markup \line {
  \overtie "overtied"
  \override #'((offset . 5) (thickness . 1))
}
\end{verbatim}
\overtie "overtied"
\override #'((offset . 1) (thickness . 5))
\overtie "overtied"

\overtied \overtied \overtied

Used properties:
- `shorten-pair ((0 . 0))`
- `height-limit (0.7)`
- `direction (1)`
- `offset (2)`
- `thickness (1)`

\replace replacements (list) arg (markup)
Used to automatically replace a string by another in the markup `arg`. Each pair of the alist `replacements` specifies what should be replaced. The key is the string to be replaced by the value string.
\markup \replace #'(("thx" . "Thanks!")) thx

Thanks!

\roman arg (markup)
Set font family to `roman`.
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}

sans serif, bold  text in roman font family  return to sans

\sans arg (markup)
Switch to the sans serif font family.
\markup {
  \default
  \hspace #2
  \sans {
    sans serif
  }
}

default  sans serif
\textbf{simple str} (string)

A simple text string; \texttt{\textbackslash markup \{ foo \}} is equivalent with \texttt{\textbackslash markup \{ \textbf{simple} \#“foo” \}}.

Note: for creating standard text markup or defining new markup commands, the use of \texttt{\textbf{simple}} is unnecessary.

\texttt{\textbackslash markup \{
    \textbf{simple} \#“simple”
    \textbf{simple} \#“text”
    \textbf{simple} \#“strings”
\}}

simple text strings

\textbf{small arg} (markup)

Set font size to -1.

\texttt{\textbackslash markup \{
    \texttt{default}
    \texttt{\hspace \#2}
    \texttt{\textbf{small}}
    \texttt{small}
\}}

\texttt{\textbf{default} \ text in small caps}

\textbf{smallCaps arg} (markup)

Emit \texttt{arg} as small caps.

Note: \texttt{\textbf{smallCaps}} does not support accented characters.

\texttt{\textbackslash markup \{
    \texttt{default}
    \texttt{\hspace \#2}
    \texttt{\textbf{smallCaps} \{
        \texttt{Text in small caps}
    \}}
\}}

\texttt{\textbf{default} \ TEXT IN SMALL CAPS}

\textbf{smaller arg} (markup)

Decrease the font size relative to the current setting.

\texttt{\textbackslash markup \{
    \texttt{\fontsize \#3.5 \{
        some large text
        \texttt{\hspace \#2}
        \texttt{\textbf{smaller} \{
            a bit smaller
        \}}
        \texttt{\hspace \#2}
        more large text
    \}}
\}}

some large text  a bit smaller  more large text
\sub arg (markup)
Set arg in subscript.
\markup {
  \concat {
    H \sub {2} \concat {0}
  }
}

\text {H}_2\text {O}

Used properties:
• font-size (0)

\super arg (markup)
Set arg in superscript.
\markup {
  E = \concat {
    mc \super {2}
  }
}

\text {E} = \text {mc}^2

Used properties:
• font-size (0)

\teeny arg (markup)
Set font size to -3.
\markup {
  \default \hspace #2 \teeny
}

\text {default teeny}

\text arg (markup)
Use a text font instead of music symbol or music alphabet font.
\markup {
  \number {
    1, 2, 
    \text {
      three, four,
    }
  }
}
\tie arg (markup)
Add a horizontal bow created with make-tie-stencil at bottom or top of \textit{arg}. Looks at \textit{thickness} to determine line thickness, and \textit{offset} to determine y-offset. The added bow fits the extent of \textit{arg}, \textit{shorten-pair} may be used to modify this. \textit{direction} may be set using an \textit{override} or direction-modifiers or \textit{voiceOne}, etc.

\markup {
\override #'(direction . 1)
\tie "above"
\override #'(direction . -1)
\tie "below"
}

\underline arg (markup)
Underline arg. Looks at \textit{thickness} to determine line thickness, \textit{offset} to determine line y-offset from \textit{arg} and \textit{underline-skip} to determine the distance of additional
lines from the others. **underline-shift** is used to get subsequent calls correct. Overriding it makes little sense, it would end up adding the provided value to the one of **offset**.

```latex
\markup \fill-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(underline-skip . 4)
  \underline \underline \underline "multiple underlined"
}\underline \underline \underline "multiple underlined"
```

Used properties:
- **underline-skip** (2)
- **underline-shift** (0)
- **offset** (2)
- **thickness** (1)

```latex
\undertie arg (markup)
\markup \line {
  \undertie "undertied"
  \override #((offset . 5) (thickness . 1))
  \undertie "undertied"
  \override #((offset . 1) (thickness . 5))
  \undertie "undertied"
}\undertie \undertie \undertie
```

Used properties:
- **shorten-pair** ((0 . 0))
- **height-limit** (0.7)
- **direction** (1)
- **offset** (2)
- **thickness** (1)

```latex
\upright arg (markup)
Set font-shape to upright. This is the opposite of italic.
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
  ```
A.11.2 Align

\center-align arg (markup)
Align arg to its X center.
\markup {
  \center-align
  \column {
    one
    two
    three
  }
}

one
two
three

\center-column args (markup list)
Put args in a centered column.
\markup {
  \center-column {
    one
    two
    three
  }
}

one
two
three

Used properties:
• baseline-skip

\column args (markup list)
Stack the markups in args vertically. The property baseline-skip determines the space between markups in args.
\markup {
  \column {
    one
    two
    three
  }
}
one
two
three

Used properties:
• baseline-skip

`\combine arg1 (markup) arg2 (markup)`
Print two markups on top of each other.
Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; for this purpose use `\overlay` instead.

\markup {
  \floattriples (0 . 4)
  \arrowhead Y #DOWN ##f
}\arrowheads.open.1M1

`\concat args (markup list)`
Concatenate args in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to "fi".

\markup {
  \concat {
    one
two
three
  }
}

onetwothree

`\dir-column args (markup list)`
Make a column of args, going up or down, depending on the setting of the `direction` layout property.

\markup {
  \override #`(direction . ,UP)
  \dir-column {
    going up
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #`(direction . 1)
  \dir-column {
    going up
  }
}

Appendix A: Notation manual tables

up up
go up
down

Used properties:
• baseline-skip
• direction

\fill-line \args (markup list)

Put markups in a horizontal line of width line-width. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

\markup \column {
  \fill-line {
    Words evenly spaced across the page
  }
  null
  \fill-line {
    \line { Text markups }
    \line {
      \italic { evenly spaced }
    }
    \line { across the page }
  }
  null
  \override #'(line-width . 50)
  \fill-line {
    Width explicitly specified
  }
}

Words evenly spaced across the page

Text markups evenly spaced across the page

Width explicitly specified

Used properties:
• line-width (#f)
• word-space (0.6)
• text-direction (1)

\fill-with-pattern space (number) dir (direction) pattern (markup) left (markup) right (markup)

Put left and right in a horizontal line of width line-width with a line of markups pattern in between. Patterns are spaced apart by space. Patterns are aligned to the dir markup.

\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  null
}
"center-aligned :"
\fill-with-pattern #1.5 #CENTER - left right
\null
"left-aligned :
\override #'(line-width . 50)
\fill-with-pattern #2 #LEFT : left first
\override #'(line-width . 50)
\fill-with-pattern #2 #LEFT : left second
}

right-aligned :
first ...................................................................................... right
second ...................................................................................... right

center-aligned :
left ..................................................................................................... right

left-aligned :

Used properties:
• line-width
• word-space

\general-align axis (integer) dir (number) arg (markup)
Align arg in axis direction to the dir side.
\markup {
\column {
one
\general-align #X #LEFT
two
three
\null
one
\general-align #X #CENTER
two
three
\null
\line {
one
\general-align #Y #UP	wo
three
}
\null
\line {
one
\general-align #Y #3.2
two
three
}
\halign \textit{dir} \ (\text{number}) \ \textit{arg} \ (\text{markup})

Set horizontal alignment. If \textit{dir} is \texttt{-1}, then it is left-aligned, while \texttt{+1} is right. Values in between interpolate alignment accordingly.

\begin{verbatim}
\markup {
\column { 
    one \halign \#LEFT 
    two 
    three 
    \null 
    one \halign \#CENTER 
    two 
    three 
    \null 
    one \halign \#RIGHT 
    two 
    three 
    \null 
    one \halign \#-5 
    two 
    three
} }
\end{verbatim}
\begin{center}
\begin{tabular}{l}
one \\
two \\
three \\
\end{tabular}
\end{center}

\hspace{amount} (number)

Create an invisible object taking up horizontal space amount.

\begin{verbatim}
\makue{one}
\hspace{#2}
\hspace{#8}
\end{verbatim}

one two three

\hcenter-in \textit{length} (number) \textit{arg} (markup)

Center \textit{arg} horizontally within a box of extending \textit{length}/2 to the left and right.

\begin{verbatim}
\new StaffGroup <<
\new Staff {
  \set Staff.instrumentName = \markup { \hcenter-in #12 Oboe \\
  c'\'1 }
}
\new Staff {
  \set Staff.instrumentName = \markup { \hcenter-in #12 Bassoon \\
  \clef tenor \\
  c'1 }
}
>>
\end{verbatim}

\begin{music}
\begin{music}
\Oboe\hspace{4}\Bassoon
\end{music}
\end{music}
\justify-field symbol (symbol)
Justify the data which has been assigned to symbol.

\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}
\markup {
  \null
}

My title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

\justify-line args (markup list)
Put markups in a horizontal line of width line-width. The markups are spread to fill the entire line and separated by equal space. If there are no arguments, return an empty stencil.

\markup {
  \justify-line {
    Space between neighboring words is constant
  }
}

Space between neighboring words is constant

Used properties:
- line-width (#f)
- word-space (0.6)
- text-direction (1)
\justify args (markup list)
Like \wordwrap, but with lines stretched to justify the margins. Use \override #'(line-width . X) to set the line width: X is the number of staff spaces.

\markup {\justify {Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.}}

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:
• text-direction (1)
• word-space
• line-width (#f)
• baseline-skip

\justify-string arg (string)
Justify a string. Paragraphs may be separated with double newlines

\markup {\override #'(line-width . 40)\justify-string #"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum"}

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
Used properties:

- `text-direction (1)`
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align arg` (markup)

Align `arg` on its left edge.

\[
\text{\texttt{\{} \texttt{\column{ \texttt{one} \texttt{\left-align \two \three} \}}} \text{\texttt{\}}}
\]

one
two
three

`\left-column args` (markup list)

Put `args` in a left-aligned column.

\[
\text{\texttt{\{} \texttt{\left-column{ \texttt{one \two \three} \}}} \text{\texttt{\}}}
\]

one
two
three

Used properties:

- `baseline-skip`

`\line args` (markup list)

Put `args` in a horizontal line. The property `word-space` determines the space between markups in `args`.

\[
\text{\texttt{\{} \texttt{\line{ \texttt{one two three} \}}} \text{\texttt{\}}}
\]

one two three

Used properties:

- `text-direction (1)`
- `word-space`
\lower amount (number) arg (markup)
Lower arg by the distance amount. A negative amount indicates raising; see also \raise.
\markup{
  \markup{\lower #3}
  two
  three
}
one  three
two

\overlay args (markup list)
Takes a list of markups combining them.
\markup{
  \fontsize #5
  \override #'(thickness . 2)
  \overlay{
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN #f
    \translate #'(0 . 4)\arrow-head #Y #UP #f
  }
}

\pad-around amount (number) arg (markup)
Add padding amount all around arg.
\markup{
  \markup{\pad-around #0.5}
  \box{
    \hspace #2
    \pad-markup padded
  }
}

\pad-markup amount (number) arg (markup)
Add space around a markup object. Identical to pad-around.
\markup{
  \markup{\pad-markup #0.5}
  \box{
    \hspace #2
    \default padded
  }
}
\pad-to-box x-ext (pair of numbers) y-ext (pair of numbers) arg (markup)
Make arg take at least x-ext, y-ext space.
\markup {\
  \box {\
    \hspace #4\
    \box {\
      \pad-to-box #'(0 . 10) #'(0 . 3) {\
        \padded\
      }\
    }\
  }\
}\default \padded

\pad-x amount (number) arg (markup)
Add padding amount around arg in the X direction.
\markup {\
  \box {\
    \hspace #4\
    \box {\
      \pad-x #2 {\
        \padded\
      }\
    }\
  }\
}\default \padded

\put-adjacent axis (integer) dir (direction) arg1 (markup) arg2 (markup)
Put arg2 next to arg1, without moving arg1.
\raise amount (number) arg (markup)
Raise arg by the distance amount. A negative amount indicates lowering, see also \lower.
The argument to \raise is the vertical displacement amount, measured in (global) staff spaces. \raise and \super raise objects in relation to their surrounding markups.
If the text object itself is positioned above or below the staff, then \texttt{\textbackslash raise} cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with \texttt{\textbackslash raise}. For vertical positioning, use the \texttt{\textbackslash padding} and/or \texttt{extra-offset} properties.

\begin{verbatim}
\markup {
  \small
  \bold
  \raise #1.0
  9/7+
}
\end{verbatim}

\begin{verbatim}
C 9/7+
\end{verbatim}

\texttt{\textbackslash right-align \texttt{arg} (markup)}
Align \texttt{arg} on its right edge.

\begin{verbatim}
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
\end{verbatim}

\begin{verbatim}
one
two
three
\end{verbatim}

\texttt{\textbackslash right-column \texttt{args} (markup list)}
Put \texttt{args} in a right-aligned column.

\begin{verbatim}
\markup {
  \right-column {
    one
two
  three
}
}
\end{verbatim}

\begin{verbatim}
one
two
three
\end{verbatim}

Used properties:
- \texttt{baseline-skip}

\texttt{\textbackslash rotate \texttt{ang} (number) \texttt{arg} (markup)}
Rotate object with \texttt{ang} degrees around its center.

\begin{verbatim}
\markup {
  default
  \hspace #2
  \rotate #45
}
\end{verbatim}
\translate offset (pair of numbers) arg (markup)
Translate arg relative to its surroundings. offset is a pair of numbers representing the displacement in the X and Y axis.
\markup {
  \* \translate #(2 . 3) \line { translated two spaces right, three up }
}
translated two spaces right, three up
*
\translate-scaled offset (pair of numbers) arg (markup)
Translate arg by offset, scaling the offset by the font-size.
\markup {
  \fontsize #5 { \* \translate #(2 . 3) \hspace #2 \* \translate-scaled #(2 . 3) \}
}

* translate *

translating the displacement in the X and Y axis.
\markup { \* \translate #(2 . 3) \}
translated two spaces right, three up
*
\translate-scaled offset (pair of numbers) arg (markup)
Translate arg by offset, scaling the offset by the font-size.
\markup { \fontsize #5 { \* \translate #(2 . 3) \hspace #2 \* \translate-scaled #(2 . 3) \}
}

* translate *

Used properties:
• font-size (0)
\vcenter arg (markup)
Align arg to its Y center.
\markup { one \vcenter two three }
one two three
\vspace amount (number)
Create an invisible object taking up vertical space of amount multiplied by 3.
\markup {
\center-column {
  one
  \vspace #2
  two
  \vspace #5
  three
}
}

one

two

three

\wordwrap-field symbol (symbol)
Wordwrap the data which has been assigned to symbol.
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat."
}
\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}
\markup {
  \null
}

My title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
\wordwrap args (markup list)

Simple wordwrap. Use \override #'(line-width . X) to set the line width, where X is the number of staff spaces.

\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
  }
}

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:
- text-direction (1)
- word-space
- line-width (#f)
- baseline-skip

\wordwrap-string arg (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

\markup {
  \override #'(line-width . 40)
  \wordwrap-string "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

  Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum"
}
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:
- text-direction (1)
- word-space
- line-width
- baseline-skip

### A.11.3 Graphic

\arrow-head \textit{axis} \textit{(integer)} \textit{dir} \textit{(direction)} \textit{filled} \textit{(boolean)}

Produce an arrow head in specified direction and axis. Use the filled head if \texttt{filled} is specified.

\begin{verbatim}
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
\end{verbatim}

\begin{verbatim}
\hspace #2
\end{verbatim}

\arrow-head \texttt{\begin{verbatim}~\end{verbatim}}

\beam \textit{width} \textit{(number)} \textit{slope} \textit{(number)} \textit{thickness} \textit{(number)}

Create a beam with the specified parameters.

\begin{verbatim}
\markup {
  \beam #5 #1 #2
}
\end{verbatim}

\begin{verbatim}
\hspace #2
\end{verbatim}

\beam \texttt{\begin{verbatim}~\end{verbatim}}

\bracket \textit{arg} \textit{(markup)}

Draw vertical brackets around \texttt{arg}.

\begin{verbatim}
\markup {
  \bracket {
    \note {2.} #UP
  }
}
\end{verbatim}

\bracket \texttt{\begin{verbatim}~\end{verbatim}}
Appendix A: Notation manual tables

\circle arg (markup)

Draw a circle around arg. Use thickness, circle-padding and font-size properties to determine line thickness and padding around the markup.

\markup {
  \circle {
    Hi
  }
}

Hi

Used properties:
- circle-padding (0.2)
- font-size (0)
- thickness (1)

\draw-circle radius (number) thickness (number) filled (boolean)

A circle of radius radius and thickness thickness, optionally filled.

\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}

Used properties:
- full-length (#t)
- phase (0)
- off (1)
- on (1)
• thickness (1)

\draw-dotted-line dest (pair of numbers)
A dotted line.
The dotted-line always extends to the whole length given by dest, without white space at beginning or end. Manual settings for off are possible to get larger or smaller space between the dots. The given (or default) value of off will be altered to fit the line-length.

\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'((thickness . 2) (off . 0.2))
  \draw-dotted-line #'(5.1 . 2.3)
}

Used properties:
• phase (0)
• off (1)
• thickness (1)

\draw-hline
Draws a line across a page, where the property span-factor controls what fraction of the page is taken up.

\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}

Used properties:
• span-factor (1)
• line-width
• draw-line-markup

\draw-line dest (pair of numbers)
A simple line.

\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}

Used properties:
• thickness (1)
\draw-squiggle-line sq-length (number) dest (pair of numbers) eq-end? (boolean)

A squiggled line.

If eq-end? is set to #t, it is ensured the squiggled line ends with a bow in same
direction as the starting one. sq-length is the length of the first bow. dest is the
end point of the squiggled line. To match dest the squiggled line is scaled accord-
ingly. Its appearance may be customized by overrides for \texttt{thickness, angularity,}
height and orientation.

\markup
\column {
\draw-squiggle-line #0.5 #'(6 . 0) ##t
\override #'(orientation . -1)
\draw-squiggle-line #0.5 #'(6 . 0) ##t
\draw-squiggle-line #0.5 #'(6 . 0) ##f
\override #'(height . 1)
\draw-squiggle-line #0.5 #'(6 . 0) ##t
\override #'(thickness . 5)
\draw-squiggle-line #0.5 #'(6 . 0) ##t
\override #'(angularity . 2)
\draw-squiggle-line #0.5 #'(6 . 0) ##t
}

\ellipse arg (markup)

Draw an ellipse around arg. Use \texttt{thickness, x-padding, y-padding} and \texttt{font-size}
properties to determine line thickness and padding around the markup.

\markup {
\ellipse {
Hi
}
}

Used properties:
\begin{itemize}
\item orientation (1)
\item height (0.5)
\item angularity (0)
\item thickness (0.5)
\end{itemize}
Appendix A: Notation manual tables

\epsfile axis (number) size (number) file-name (string)
   In-line an EPS image. The image is scaled along axis to size.
   \markup {
      \general-align #Y #DOWN {
         \epsfile #X #20 "context-example.eps"
         \epsfile #Y #20 "context-example.eps"
      }
   }

\filled-box xext (pair of numbers) yext (pair of numbers) blot (number)
   Draw a box with rounded corners of dimensions xext and yext. For example,
   \filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
   creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to
   1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).
   \markup {
      \filled-box #'(0 . 4) #'(0 . 4) #0
      \filled-box #'(0 . 2) #'(-4 . 2) #0.4
      \filled-box #'(1 . 8) #'(0 . 7) #0.2
      \with-color #white
         \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
   }

\hbracket arg (markup)
   Draw horizontal brackets around arg.
   \markup {
      \hbracket {\line {one two three}}
   }

\oval arg (markup)
   Draw an oval around arg. Use thickness, x-padding, y-padding and font-size
   properties to determine line thickness and padding around the markup.
   \markup {
   }
\oval {
  Hi
}

Hi

Used properties:
• y-padding (0.75)
• x-padding (0.75)
• font-size (0)
• thickness (1)

\parenthesize arg (markup)
Draw parentheses around arg. This is useful for parenthesizing a column containing several lines of text.

\markup {
  \parenthesize 
  \column {
    foo
    bar
  }
  \override #'(angularity . 2)
  \parenthesize 
  \column {
    bah
    baz
  }
}

\path thickness (number) commands (list)
Draws a path with line thickness according to the directions given in commands. commands is a list of lists where the car of each sublist is a drawing command and the cdr comprises the associated arguments for each command.

There are seven commands available to use in the list commands: moveto, rmoveto, lineto, rlineto, curveto, rcurveto, and closepath. Note that the commands that begin with r are the relative variants of the other three commands.

The commands moveto, rmoveto, lineto, and rlineto take 2 arguments; they are the X and Y coordinates for the destination point.
The commands \texttt{curveto} and \texttt{rcurveto} create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The \texttt{closepath} command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands \textit{must} begin with a \texttt{moveto} or \texttt{rmoveto} to work with the SVG output.

Line-cap styles and line-join styles may be customized by overriding the \texttt{line-cap-style} and \texttt{line-join-style} properties, respectively. Available line-cap styles are 'butt, 'round, and 'square. Available line-join styles are 'miter, 'round, and 'bevel.

The property \texttt{filled} specifies whether or not the path is filled with color.

\begin{verbatim}
samplePath = '#((moveto 0 0)
    (lineto -1 1)
    (lineto 1 1)
    (lineto 1 -1)
    (curveto -5 -5 -5 5 -1 0)
    (closepath))

\markup {
  \path #0.25 #samplePath
  \override #'(line-join-style . miter) \path #0.25 #samplePath
  \override #'(filled . #t) \path #0.25 #samplePath
}
\end{verbatim}

Used properties:
- \texttt{filled} (#f)
- \texttt{line-join-style} (round)
- \texttt{line-cap-style} (round)

\texttt{\postscript str} (string)
This inserts \texttt{str} directly into the output as a PostScript command string.

\begin{verbatim}
ringsps = 
  0.15 setlinewidth  
  0.9 0.6 moveto  
  0.4 0.6 0.5 0 361 arc stroke  
  1.0 0.6 0.5 0 361 arc stroke  
" 

rings = \markup {
  \with-dimensions 
  \postscript #ringsps
}
\end{verbatim}
\rounded-box { arg }

Draw a box with rounded corners around \textit{arg}. Looks at \texttt{thickness}, \texttt{box-padding} and \texttt{font-size} properties to determine line thickness and padding around the markup; the \texttt{corner-radius} property makes it possible to define another shape for the corners (default is 1).

\scale { factor-pair } { arg }

Scale \textit{arg}. \texttt{factor-pair} is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

\triangle { filled }

A triangle, either filled or empty.
\[ \triangle \#t \hspace{#2} \triangle \#f \]

\begin{itemize}
\item baseline-skip (2)
\item font-size (0)
\item thickness (0.1)
\end{itemize}

\texttt{\with-url url (string) arg (markup)}

Add a link to URL \textit{url} around \textit{arg}. This only works in the PDF backend.

\begin{verbatim}
\markup {
  \with-url "https://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
\end{verbatim}

LilyPond ... \textit{music notation for everyone}

\section*{A.11.4 Music}

\texttt{\compound-meter time-sig (number or pair)}

Draw a numeric time signature.

\begin{verbatim}
\markup {
  \column {
    \line { Single number: \compound-meter #3 }
    \line { Conventional: \compound-meter #'(4 . 4)
      or \compound-meter #'(4 4) }
    \line { Compound: \compound-meter #'(2 3 8) }
    \line { Single-number compound: \compound-meter #'((2) (3)) }
    \line { Complex compound: \compound-meter #'((2 3 8) (3 4)) }
  }
}
\end{verbatim}

Single number: 3
Conventional: \textfrac{4}{4} or \textfrac{4}{4}
Compound: \textfrac{2+3}{8}
Single-number compound: \textfrac{2+3}{8}
Complex compound: \textfrac{2+3}{8} + \textfrac{3}{4}

\texttt{\customTabClef num-strings (integer) staff-space (number)}

Draw a tab clef sans-serif style.

\texttt{\doubleflat}

Draw a double flat symbol.

\begin{verbatim}
\markup {
}
\end{verbatim}
\doubleflat
}
\bflat
\doubleslash

Draw a double sharp symbol.
\markup {
  \doubleslash
}
\doublesharp

Create a fermata glyph. When direction is DOWN, use an inverted glyph. Note that within music, one would usually use the \fermata articulation instead of a markup.
\markup {
  \fermata
  \override #`(direction . ,DOWN) \fermata
}
\fermata

\flat

Draw a flat symbol.
\markup {
  \flat
}
\flat

\musicglyph glyph-name (string)

glyph-name is converted to a musical symbol; for example, \musicglyph #$"accidentals.natural" selects the natural sign from the music font. See Section “The Emmentaler font” in Notation Reference for a complete listing of the possible glyphs.
\markup {
  \musicglyph #$"f"
  \musicglyph #$"rests.2"
  \musicglyph #$"clefs.G_change"
}
\musicglyph

\natural

Draw a natural symbol.
\markup {
\natural
\natural

\note-by-number log (number) dot-count (number) dir (number)
Construct a note symbol, with stem and flag. By using fractional values for \textit{dir},
longer or shorter stems can be obtained. Supports all note-head-styles. Ancient note-head-styles will get mensural-style-flags. \textit{flag-style} may be over-ridden independently. Supported flag-styles are default, old-straight-flag, modern-straight-flag, flat-flag, mensural and neomensural. The latter two flag-styles will both result in mensural-flags. Both are supplied for convenience.

\markup {
\note-by-number #3 #0 #DOWN
\hspace #2
\note-by-number #1 #2 #0.8
}

\note \d\.

Used properties:
• style ('()')
• flag-style ('()')
• font-size (0)

\note duration (duration) dir (number)
This produces a note with a stem pointing in \textit{dir} direction, with the \textit{duration} for the
note head type and augmentation dots. For example, \textbackslash note {4.} $\#-$0.75 creates a
dotted quarter note, with a shortened down stem.

\markup {
\override #'(style . cross)
\note {4.} #UP
\hspace #2
\note {$\breve$} #0
}

\note \d\.

Used properties:
• style ('()')
• flag-style ('()')
• font-size (0)

\rest-by-number log (number) dot-count (number)
A rest or multi-measure-rest symbol.

\markup {
\rest-by-number #3 #2
\hspace #2
\rest-by-number #0 #1
Used properties:
- \texttt{multi-measure-rest} (#f)
- style ('())
- font-size (0)

\textbf{\texttt{\textbackslash rest duration (string)}}

This produces a rest, with the \texttt{duration} for the rest type and augmentation dots.
"breve", "longa" and "maxima" are valid input-strings.

Printing MultiMeasureRests could be enabled with \texttt{\textbackslash override \#'(multi-measure-rest . #t)} If MultiMeasureRests are taken, the MultiMeasureRestNumber is printed above. This is enabled for all styles using default-glyphs. Could be disabled with \texttt{\textbackslash override \#'(multi-measure-rest-number . #f)}

```
\markup {
  \score {
    \new PianoStaff <<
    \new Staff \relative c' {
      \key f \major
      \time 3/4
    }
  }
}
```
\mark \markup { Allegro }
f2\p( a4)
c2( a4)
bes2( g'4)
f8( e) e4 r
}\new Staff \relative c {
\clef bass
\key f \major
\time 3/4
f8( a c a c a
f c' es c es c)
f,( bes d bes d bes)
f( g bes g bes g)
}
\layout {
indent = 0.0\cm
\context {\Score
\override RehearsalMark.break-align-symbols =
\markup {#'(time-signature key-signature)}
\override RehearsalMark.self-alignment-X = #LEFT
}
\context {\Staff
\override TimeSignature.break-align-anchor-alignment =
\markup {#LEFT}
}
}
\semiflat
\draw a semiflat symbol.
\markup {
\semiflat
}\d
\semisharp
Draw a 3/2 flat symbol.
\markup {
  \semisharp
}

\sesquiflat
Draw a \ markup \ \{ \ \sesquiflat \}

\sesquisharp
Draw a 3/2 sharp symbol.
\markup {\sesquisharp}

\sharp
Draw a sharp symbol.
\markup {\sharp}

\tied-lyric \ string \ (string)
Like simple-markup, but use tie characters for ‘~’ tilde symbols.
\markup \column {\tied-lyric Siam navi\-all'onde\-algenti Lasciate\-in abbandono
\tied-lyric Impetuosi venti I nostri\-affetti sono
\tied-lyric Ogni diletto\-e scoglio Tutta la vita\-e\-un mar.}

Siam navi all'onde algenti Lasciate in abbandono
Impetuosi venti I nostri affetti sono
Ogni diletto e scoglio Tutta la vita e un mar.

Used properties:
- word-space
A.11.5 Instrument Specific Markup

\fret-diagram definition-string (string)

Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for definition-string:
- Diagram items are separated by semicolons.
- Possible items:
  - s: number – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - t: number – Set the line thickness (relative to normal line thickness). Default: 0.5.
  - h: number – Set the height of the diagram in frets. Default: 4.
  - f: number – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
  - d: number – Set radius of dot, in terms of fret spacing. Default: 0.25.
  - p: number – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
  - c: string1-string2-fret – Include a barre mark from string1 to string2 on fret.
  - string-fret – Place a dot on string at fret. If fret is ‘o’, string is identified as open. If fret is ‘x’, string is identified as muted.
  - string-fret-fingering – Place a dot on string at fret, and label with fingering as defined by the f: code.
- Note: There is no limit to the number of fret indications per string.

Used properties:
- thickness (0.5)
- fret-diagram-details
- size (1.0)
- align-dir (-0.4)

\fret-diagram-terse definition-string (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for definition-string:
- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by ‘x’.
- Open strings are indicated by ‘o’.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
• Fingerings are given by following the fret number with a −, followed by the finger indicator, e.g. ‘3-2’ for playing the third fret with the second finger.

• Where a barre indicator is desired, follow the fret (or fingering) symbol with -( to start a barre and −) to end the barre.

Used properties:
• thickness (0.5)
• fret-diagram-details
• size (1.0)
• align-dir (-0.4)

\fret-diagram-verbose marking-list (pair)
Make a fret diagram containing the symbols indicated in marking-list. For example,
\markup \fret-diagram-verbose
#'(mute 6) (mute 5) (open 4)
(place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
produces a standard D chord diagram without fingering indications.

Possible elements in marking-list:

*(mute string-number)*
Place a small ‘x’ at the top of string string-number.

*(open string-number)*
Place a small ‘o’ at the top of string string-number.

*(barre start-string end-string fret-number)*
Place a barre indicator (much like a tie) from string start-string to string end-string at fret fret-number.

*(capo fret-number)*
Place a capo indicator (a large solid bar) across the entire fretboard at fret location fret-number. Also, set fret fret-number to be the lowest fret on the fret diagram.

*(place-fret string-number fret-number [finger-value] [color-modifier] [color] ['parenthesized ['default-paren-color]]))*
Place a fret playing indication on string string-number at fret fret-number with an optional fingering label finger-value, an optional color modifier color-modifier, an optional color color, an optional parenthesis 'parenthesized and an optional parenthesis color 'default-paren-color. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable dot-color or for a single dot by setting the value of color. The dot can be parenthesized by adding 'parenthesized. By default the color for the parenthesis is taken from the dot. Adding 'default-paren-color will take the parenthesis-color from the global dot-color, as a fall-back black will be used. Setting color-modifier to inverted inverts the dot color for a specific fingering. The values for string-number, fret-number, and the optional finger should be entered first in that order. The order of the other optional arguments does not matter. If the finger part of the place-fret element is present, finger-value will be displayed according to the setting of the variable finger-code. There is no limit to the number of fret indications per string.
Appendix A: Notation manual tables

Used properties:
- thickness (0.5)
- fret-diagram-details
- size (1.0)
- align-dir (-0.4)

\harp-pedal definition-string (string)
Make a harp pedal diagram.
Possible elements in definition-string:
- pedal is up
- pedal is neutral
v pedal is down
l vertical divider line
o the following pedal should be circled (indicating a change)

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you’ll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (\override Voice.TextScript.size = #0.3) for the overall, the thickness property (\override Voice.TextScript.thickness = #3) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (\override Voice.TextScript.harp-pedal-details.box-width = #1). It contains the following settings: box-offset (vertical shift of the box center for up/down pedals), box-width, box-height, space-before-divider (the spacing between two boxes before the divider) and space-after-divider (box spacing after the divider).

\markup \harp-pedal #"^-v|--ov^"

Used properties:
- thickness (0.5)
- harp-pedal-details (')
- size (1.2)

\woodwind-diagram instrument (symbol) user-draw-commands (list)
Make a woodwind-instrument diagram. For example, say

\markup \woodwind-diagram
  #'oboe #'(lh . (d ees)) (cc . (five3qT1q)) (rh . (gis)))
for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:
- piccolo
Appendix A: Notation manual tables

- flute
- oboe
- clarinet
- bass-clarinet
- saxophone
- bassoon
- contrabassoon

To see all of the callable keys for a given instrument, include the function (print-keys 'instrument) in your .ly file, where instrument is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

- 1q (1/4 covered)
- 1h (1/2 covered)
- 3q (3/4 covered)
- R (ring depressed)
- F (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, three3qTR effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, threeRT effectuates a trill between R and open, whereas threeTR effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke (print-keys-verbose 'instrument).

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

\markup \woodwind-diagram #'oboe #'()

Used properties:

- graphical (#t)
- thickness (0.1)
- size (1)

A.11.6 Accordion Registers

\discant name (string)

\discant name generates a discant accordion register symbol.

To make it available,

#(use-modules (scm accreg))

is required near the top of your input file.

The register names in the default \discant register set have modeled after numeric Swiss notation like depicted in http://de.wikipedia.org/wiki/Register_%28Akkordeon%29, omitting the slashes and dropping leading zeros.

The string name is basically a three-digit number with the lowest digit specifying the number of 16’ reeds, the tens the number of 8’ reeds, and the hundreds specifying the number of 4’ reeds. Without modification, the specified number of reeds in 8’ is centered in the symbol. Newer instruments may have registrations where 8’ can be used either within or without a tone chamber, ‘cassotto’. Notationally, the central dot then indicates use of cassotto. One can suffix the tens’ digits ‘1’ and ‘2’ with
‘+’ or ‘-’ to indicate clustering the dots at the right or left respectively rather than centered.

Some examples are

\[ \text{\textbackslash discant "1" \quad \text{\textbackslash discant "1+0"}} \]

\[ \text{\textbackslash discant "120" \quad \text{\textbackslash discant "131"}} \]

Used properties:
• font-size (0)

\[ \text{\textbackslash freeBass \textit{name} \ (\text{string})} \]

\[ \text{\textbackslash freeBass \textit{name}} \text{\ generates a free bass/} \text{converter accordion register symbol for the usual two-reed layout.} \]

To make it available,

\[ \#(\text{use-modules \ (scm \ accreg)}) \]

is required near the top of your input file.

Available registrations are

\[ \text{\textbackslash freeBass "1" \quad \text{\textbackslash freeBass "11"}} \]

\[ \text{\textbackslash freeBass "10"} \]

Used properties:
• font-size (0)

\[ \text{\textbackslash stdBass \textit{name} \ (\text{string})} \]

\[ \text{\textbackslash stdBass \textit{name}} \text{\ generates a standard bass accordion register symbol.} \]

To make it available,

\[ \#(\text{use-modules \ (scm \ accreg)}) \]

is required near the top of your input file.

The default bass register definitions have been modeled after the article [http://www.accordions.com/index/art/stradella.shtml](http://www.accordions.com/index/art/stradella.shtml) originally appearing in Accord Magazine.

The underlying register model is
This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five reed definitions makes more sense than using a four reed layout: in that manner, the ‘Master’ register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are

- \stdBass "Soprano"  
- \stdBass "Soft Bass"

- \stdBass "Alto"  
- \stdBass "Soft Tenor"

- \stdBass "Tenor"  
- \stdBass "Bass/Alto"

- \stdBass "Master"

Used properties:
- font-size (0)

\stdBassIV name (string)
\stdBassIV name generates a standard bass accordion register symbol.

To make it available,

#(use-modules (scm accreg))

is required near the top of your input file.

The main use is for four-reed standard bass instruments with reedbank layout

Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches ‘Tenor’, ‘Master’, ‘Soft Bass’. Note that the sound is quite darker than the same registrations of ‘c,’-based instruments.

Available registrations are
Appendix A: Notation manual tables

\stdBassIV "Soprano" \stdBassIV "Soft Bass"
\stdBassIV "Alto" \stdBassIV "Bass/Alto"
\stdBassIV "Tenor" \stdBassIV "Soft Bass/Alto"
\stdBassIV "Master" \stdBassIV "Soft Tenor"

Used properties:

- font-size (0)

\stdBassV name (string)
\stdBassV name generates a standard bass accordion register symbol.

To make it available,

###(use-modules (scm accreg))

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout

This tends to be the bass layout for Hohner’s Morino series without convertor or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. The Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional ‘Solo Bass’ setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are
# stdBassVI name (string)

stdBassVI \texttt{name} generates a standard bass accordion register symbol for six reed basses.

To make it available, 

\#(use-modules (scm accreg))

is required near the top of your input file.

This is primarily the register layout for the Hohner “Gola” model. The layout is

The registers are effectively quite similar to that of \texttt{stdBass}. An additional bass reed at alto pitch is omitted for esthetical reasons from the ‘Master’ setting, so the symbols are almost the same except for the ‘Alto/Soprano’ register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are
A.11.7 Other

\auto-footnote \textit{mkup} (markup) \textit{note} (markup)

Have footnote \textit{note} act as an annotation to the markup \textit{mkup}.

\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}

\hspace #2
\backslashed-digit #7

\backslashed-digit \num {5}

\char \num {5}

A feta number, with backslash. This is for use in the context of figured bass notation.

\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}

\char \num {5}

Produce a single character. Characters encoded in hexadecimal format require the prefix \#x.

\markup {
  \char \num {5}
}
\char #65 \char #x00a9

A ©

\eyeglasses
Prints out eyeglasses, indicating strongly to look at the conductor.
\markup { \eyeglasses }

\first-visible args (markup list)
Use the first markup in args that yields a non-empty stencil and ignore the rest.
\markup { \first-visible { \fromproperty #'header:composer \italic Unknown } }

\footnote mkup (markup) note (markup)
Have footnote note act as an annotation to the markup mkup.
\markup { \auto-footnote a b \override #'(padding . 0.2) \auto-footnote c d } 
  a c

The footnote will not be annotated automatically.

\fraction arg1 (markup) arg2 (markup)
Make a fraction of two markups.
\markup { \pi \approx \fraction 355 113 } 
\[ \pi \approx \frac{355}{113} \]

Used properties:
• font-size (0)

\fromproperty symbol (symbol)
Read the symbol from property settings, and produce a stencil from the markup contained within. If symbol is not defined, it returns an empty markup.
\header { myTitle = "myTitle" title = \markup { } }
\left-brace \textit{size} \ (\textit{number})
\ A feta brace in point size \textit{size}.
\markup {\left-brace #35 \hspace #2 \left-brace #45}

\lookup \textit{glyph-name} \ (\textit{string})
\ Lookup a glyph by name.
\markup {\override #'(font-encoding . fetaBraces) {\lookup "brace200" \hspace #2 \rotate #180 \lookup "brace180"}}

\markalphabet \textit{num} \ (\textit{integer})
\ Make a markup letter for \textit{num}. The letters start with A to Z and continue with double letters.
\markup {\markalphabet #8 \hspace #2 \markalphabet #26}
\texttt{\textbf{H} \ Z}

\texttt{\textbackslash markletter \textit{num}} (\texttt{integer})

Make a markup letter for \texttt{num}. The letters start with A to Z (skipping letter I), and continue with double letters.

\texttt{\textbackslash markup \{}
\texttt{\markletter \#8}
\texttt{\hspace \#2}
\texttt{\markletter \#26}
\texttt{\}}

\texttt{\textbf{H} \ AA}

\texttt{\textbackslash null}

An empty markup with extents of a single point.

\texttt{\textbackslash markup \{}
\texttt{\null}
\texttt{\}}

\texttt{\textbackslash on-the-fly procedure (procedure) \texttt{arg (markup)}}

Apply the \texttt{procedure} markup command to \texttt{arg}. \texttt{procedure} takes the same arguments as \texttt{interpret-markup} and returns a stencil.

\texttt{\textbackslash override new-prop (pair) \texttt{arg (markup)}}

Add the argument \texttt{new-prop} to the property list. Properties may be any property supported by Section “font-interface” in \textit{Internals Reference}, Section “text-interface” in \textit{Internals Reference} and Section “instrument-specific-markup-interface” in \textit{Internals Reference}.

\texttt{new-prop} may be either a single alist pair, or non-empty alist of its own.

\texttt{\textbackslash markup \{}
\texttt{\undertie "undertied"}
\texttt{\override \#'(offset . 15)}
\texttt{\undertie "offset undertied"}
\texttt{\override \#'(offset . 15)(thickness . 3))}
\texttt{\undertie "offset thick undertied"}
\texttt{\}}

\texttt{\begin{center}\underline{\texttt{undertied}} \underline{\texttt{offset undertied}} \underline{\texttt{offset thick undertied}}\end{center}}

\texttt{\textbackslash page-link page-number (number) \texttt{arg (markup)}}

Add a link to the page \texttt{page-number} around \texttt{arg}. This only works in the PDF backend.

\texttt{\textbackslash markup \{}
\texttt{\page-link \#2 \{ \italic \{ This links to page 2... \} \}}
\texttt{\}}

\textit{This links to page 2...}
Appendix A: Notation manual tables

\page-ref label (symbol) gauge (markup) default (markup)
Reference to a page number. label is the label set on the referenced page (using \label or \tocItem), gauge a markup used to estimate the maximum width of the page number, and default the value to display when label is not found.
(If the current book or bookpart is set to use roman numerals for page numbers, the reference will be formatted accordingly – in which case the gauge’s width may require additional tweaking.)

\pattern count (non-negative integer) axis (non-negative integer) space (number) pattern (markup)
Prints count times a pattern markup. Patterns are spaced apart by space (defined as for \hspace or \vspace, respectively). Patterns are distributed on axis.
\markup \column {
   "Horizontally repeated :"
   \pattern #7 #X #2 \flat
   \null
   "Vertically repeated :"
   \pattern #3 #Y #0.5 \flat
}
Horizontally repeated :
\flat \flat \flat \flat \flat \flat \flat
Vertically repeated :
\flat
\flat
\flat

\property-recursive symbol (symbol)
Print out a warning when a header field markup contains some recursive markup definition.

\right-brace size (number)
A feta brace in point size size, rotated 180 degrees.
\markup {
   \right-brace #45
   \hspace #2
   \right-brace #35
}

\slashed-digit num (integer)
A feta number, with slash. This is for use in the context of figured bass notation.
\markup {
   \slashed-digit #5
   \hspace #2
   \override #'(thickness . 3)
\slashed-digit #7
}

5 7

Used properties:
• thickness (1.6)
• font-size (0)

\stencil stil (stencil)
Use a stencil as markup.
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}

\strut
Create a box of the same height as the space in the current font.

\transparent arg (markup)
Make arg transparent.
\markup {
  \transparent {
    invisible text
  }
}

\verbatim-file name (string)
Read the contents of file name, and include it verbatim.
\markup {
  \verbatim-file #'"simple.ly"
}

%%% A simple piece in LilyPond, a scale.
\relative {
  c' d e f g a b c
}
%%% Optional helper for automatic updating by convert-ly.
%%% May be omitted.
\version "2.19.21"

\whiteout arg (markup)
Provide a white background for arg. The shape of the white background is determined by style. The default is box which produces a rectangle. rounded-box produces a rounded rectangle. outline approximates the outline of the markup.
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
Appendix A: Notation manual tables

\绺 \texttt{(thickness . 1.5)}
\texttt{\textcolor{whiteout} whiteout-box}

\绺 \texttt{\textcolor{whiteout} whiteout-box}

\绺 \texttt{\textcolor{whiteout} whiteout-rounded-box}

\绺 \texttt{\textcolor{whiteout} whiteout-outline}

\textbf{Used properties:}
\begin{itemize}
\item \texttt{thickness (())}
\item \texttt{style (box)}
\end{itemize}

\texttt{\textcolor{whiteout} with-color color (color) arg (markup)}

\textit{Draw arg in color specified by color.}

\绺 \texttt{\textcolor{whiteout} \texttt{\textcolor{red} red}}
\texttt{\textcolor{whiteout} \hspace #2}
\texttt{\textcolor{whiteout} \texttt{\textcolor{green} green}}
\texttt{\textcolor{whiteout} \hspace #2}
\texttt{\textcolor{whiteout} \texttt{\textcolor{"#0000ff} blue}}

\绺 \texttt{red green blue}

\texttt{\textcolor{whiteout} with-dimensions-from arg1 (markup) arg2 (markup)}

\textit{Print arg2 with the dimensions of arg1.}

\texttt{\textcolor{whiteout} with-dimensions x (pair of numbers) y (pair of numbers) arg (markup)}

\textit{Set the dimensions of arg to x and y.}
Appendix A: Notation manual tables

\with-link label (symbol) arg (markup)
Add a link to the page holding label label around arg. This only works in the PDF backend.
\markup {
  \with-link #'label {
    \italic { This links to the page containing the label... }
  }
}
This links to the page containing the label...

\with-outline outline (markup) arg (markup)
Print arg with the outline and dimensions of outline.

A.12 Text markup list commands
The following commands can all be used with \markuplist:

\column-lines args (markup list)
Like \column, but return a list of lines instead of a single markup. baseline-skip determines the space between each markup in args.
Used properties:
  • baseline-skip

\justified-lines args (markup list)
Like \justify, but return a list of lines instead of a single markup. Use \override-lines #'(line-width . X) to set the line width; X is the number of staff spaces.
Used properties:
  • text-direction (1)
  • word-space
  • line-width (#f)
  • baseline-skip

\map-markup-commands compose (procedure) args (markup list)
This applies the function compose to every markup in args (including elements of markup list command calls) in order to produce a new markup list. Since the return value from a markup list command call is not a markup list but rather a list of stencils, this requires passing those stencils off as the results of individual markup calls. That way, the results should work out as long as no markups rely on side effects.

\override-lines new-prop (pair) args (markup list)
Like \override, for markup lists.

\score-lines score (score)
This is the same as the \score markup but delivers its systems as a list of lines. Its score argument is entered in braces like it would be for \score.

\table column-align (number list) lst (markup list)
Returns a table.
  column-align specifies how each column is aligned, possible values are -1, 0, 1. The number of elements in column-align determines how many columns will be printed.
The entries to print are given by lst, a markup-list. If needed, the last row is filled
up with point-stencils. Overriding padding may be used to increase columns horizontal distance. Overriding baseline-skip to increase rows vertical distance.

\markuplist {
\override #(padding . 2)
\table
#'(0 1 0 -1)
{
\underline { center-aligned right-aligned center-aligned left-aligned }
one \number 1 thousandth \number 0.001
eleven \number 11 hundredth \number 0.01
twenty \number 20 tenth \number 0.1
thousand \number 1000 one \number 1.0
}
}

center-aligned right-aligned center-aligned left-aligned

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td>1</td>
<td>thousandth</td>
<td>0.001</td>
</tr>
<tr>
<td>eleven</td>
<td>11</td>
<td>hundredth</td>
<td>0.01</td>
</tr>
<tr>
<td>twenty</td>
<td>20</td>
<td>tenth</td>
<td>0.1</td>
</tr>
<tr>
<td>thousand</td>
<td>1000</td>
<td>one</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Used properties:
- baseline-skip
- padding (0)

\table-of-contents
Used properties:
- baseline-skip

\wordwrap-internal justify (boolean) args (markup list)
Internal markup list command used to define \justify and \wordwrap.
Used properties:
- text-direction (1)
- word-space
- line-width (#f)

\wordwrap-lines args (markup list)
Like \wordwrap, but return a list of lines instead of a single markup. Use \override-lines #'(line-width . X) to set the line width, where X is the number of staff spaces.
Used properties:
- text-direction (1)
- word-space
- line-width (#f)
Appendix A: Notation manual tables

• baseline-skip
\wordwrap-string-internal justify (boolean) arg (string)
Internal markup list command used to define \justify-string and \wordwrap-string.
Used properties:
• text-direction (1)
• word-space
• line-width

A.13 List of special characters
The following special characters references can be used; for more details, see [ASCII aliases], page 526.
The HTML syntax is used and most of these references are the same as HTML. The rest of them are inspired by \LaTeX.
The characters are boxed so that you can see their size. A small padding has been added between the character and the box for more readability.

&iexcl; | &iquest; | § &solidus; | ¤ &flq; | ␡
&frq; | ¤ &flqq; | ␢ &frqq; | ¢ &glq; | ␢
&grq; | ¢ &glqq; | ␣ &grqq; | ¨ &elq; | ␣
&erq; | ¨ &elqq; | ␤ &erqq; | ␤ &ensp; | ␤
emsp; | &thinsp; | . &nbsp; | ␥ &nnbsp; | ␥
&zwj; | . &zwnj; | . &middot; | &bull; | ␦
&copyright; | © &registered; | ™ &trademark; | † &dagger; | ␠
&Dagger; | ™ &numero; | ® &ordf; | ® &ordm; | ␠
&para; | ℠ &sect; | ° &deg; | ° &numero; | ℠
&permil; | ℠ &brvbar; | † &acute; | &acutedbl; | ␠
&grave; | &breve; | &caron; | &cedilla; | ␠
&circumflex; | &diaeresis; | &macron; | &aa; | ␢
&AA; | &ae; | &AE; | &auml; | ␢
A.14 List of articulations

In LilyPond’s internal logic, an ‘articulation’ is any object (other than dynamics) that may be attached directly after a rhythmic event: notes, chords; even silences and skips, or the empty chord construct <>. Even slurs, fingerings and text scripts are technically articulations, although these are not shown here. Therefore, the following lists include not only articulation marks, but also all other scripts in the Emmentaler font that may be attached to notes (the way an accent is entered as ‘c’\accent’ or ‘c’–>’). Each example shows the script in its two possible vertical positions: respectively up and down, as well as its default (neutral) position. See also [Script glyphs], page 693, for a more extensive list of glyphs, for use with the \musicglyph markup command as explained in [Music notation inside markup], page 264.

Articulation scripts
Ornament scripts

\prall
\prallup
\pralldown
\upprall
\downprall
\prallprall
\lineprall
\prallmordent
\mordent
\upmordent
\downmordent
\trill
\turn
\reverseturn
\slashturn
\haydnturn

Fermata scripts

\veryshortfermata
\shortfermata
\fermata
\longfermata
\verylongfermata
\henzeshortfermata
\henzelongfermata

Instrument-specific scripts

\upbow
\downbow
\flageolet
\open
Appendix A: Notation manual tables

Repeat sign scripts

Ancient scripts

A.15 Percussion notes

<table>
<thead>
<tr>
<th>Percussion note</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>bassdrum</td>
<td>\halfopen</td>
</tr>
<tr>
<td>acousticbassdrum</td>
<td>\lfoot</td>
</tr>
<tr>
<td>snare</td>
<td>\rfoot</td>
</tr>
<tr>
<td>electricsnare</td>
<td>\rtoe</td>
</tr>
<tr>
<td>lowfloortom</td>
<td>\snappizzicato</td>
</tr>
<tr>
<td>highfloortom</td>
<td>\stoppedor++</td>
</tr>
<tr>
<td>lowtom</td>
<td>\thumb</td>
</tr>
<tr>
<td>bd</td>
<td>\lheel</td>
</tr>
<tr>
<td>bda</td>
<td>\rheel</td>
</tr>
<tr>
<td>sn</td>
<td>\ltoe</td>
</tr>
<tr>
<td>sna</td>
<td>\rtoe</td>
</tr>
<tr>
<td>sne</td>
<td>\segno</td>
</tr>
<tr>
<td>tomfl</td>
<td>\coda</td>
</tr>
<tr>
<td>tomfh</td>
<td>\varcoda</td>
</tr>
<tr>
<td>toml</td>
<td>\accentus</td>
</tr>
<tr>
<td></td>
<td>\circulus</td>
</tr>
<tr>
<td></td>
<td>\ictus</td>
</tr>
<tr>
<td></td>
<td>\semicirculus</td>
</tr>
<tr>
<td></td>
<td>\signumcongruentiae</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

hightom
  tomh

lowmidtom
  toml

himidtom
  tommh

highhat
  hh

closedhihat
  hhc

openhighhat
  hho

halfopenhihat
  hhho

pedalhihat
  hhp

-crashcymbal
  cymc

-crashcymbala
  cymca

-crashcymbalb
  cymcb

-ridecymbal
  cymr

-ridecymbala
  cymra

-ridecymbalb
  cymrb

chinesecymbal
  cymch

splashcymbal
  cyms

ridebell
  rb

cowbell
  cb

hibongo
  boh

openhibongo
  boho

mutehibongo
  bohm

lobongo
  bol

openlobongo
  bolo

mutelobongo
  bolm

hitimbale
  timh

lotimbale
  timl

hiconga
  cgh

openhiconga
  cgho

mutehiconga
  cghm

loconga
  cgl

openloconga
  cglo

muteloconga
  cglm

hitimbale
  timh

lotimbale
  timl
A.16 Technical glossary

A glossary of the technical terms and concepts used internally in LilyPond. These terms may appear in the manuals, on mailing lists or in the source code.

alist

An association list or **alist** for short is a Scheme pair which associates a value with a key: `(key . value)`. For example, in `scm/lily.scm`, the alist “type-p-name-alist” associates certain type predicates (e.g., `ly:music?`) with names (e.g., “music”) so that type-check failures can be reported with a console message that includes the name of the expected type predicate.

callback

A **callback** is a routine, function or method whose reference is passed as an argument in a call to another routine, so allowing the called routine to invoke it. The technique enables a lower-level software layer to call a function defined in a higher layer. Callbacks are used extensively in LilyPond to permit user-level Scheme code to define how many low-level actions are performed.

closure

In Scheme, a **closure** is created when a function, usually a lambda expression, is passed as a variable. The closure contains the function’s code plus references to the lexical bindings of the function’s free variables (i.e., those variables used in the expression but defined outside it). When this function is applied to different arguments later, the free variable bindings that were captured in the closure are used to obtain the values of the free variables to be used in the calculation. One useful property of closures is the retention of internal variable values between invocations, so permitting state to be maintained.

glyph

A **glyph** is a particular graphical representation of a typographic character, or a combination of two characters forming a ligature. A set of glyphs with a single style and shape comprise a font, and a set of fonts covering several styles and sizes comprise a typeface.

See also

Notation Reference: Section 1.8.3 [Fonts], page 267, Section 3.3.3 [Special characters], page 524.

grob

LilyPond objects which represent items of notation in the printed output such as note heads, stems, slurs, ties, fingering, clefs, etc are called ‘Layout objects’, often known as ‘GRectical ObJects’, or **grobs** for short. They are represented by instances of the **Grob** class.

See also

immutable

An immutable object is one whose state cannot be modified after creation, in contrast to a mutable object, which can be modified after creation.

In LilyPond, immutable or shared properties define the default style and behavior of grobs. They are shared between many objects. In apparent contradiction to the name, they can be changed using \override and \revert.

See also

Notation Reference: [mutable], page 774.

interface

Actions and properties which are common to a number of grobs are grouped together in an object called a grob-interface, or just ‘interface’ for short.

See also


Notation Reference: Section 5.2.2 [Layout interfaces], page 617.

Internals Reference: Section “Graphical Object Interfaces” in Internals Reference.

lexer

A lexer is a program which converts a sequence of characters into a sequence of tokens, a process called lexical analysis. The LilyPond lexer converts the stream obtained from an input .ly file into a tokenized stream more suited to the next stage of processing - parsing, for which see [parser], page 774. The LilyPond lexer is built with Flex from the lexer file lily/lexer.ll which contains the lexical rules. This file is part of the source code and is not included in the LilyPond binary installation.

mutable

A mutable object is one whose state can be modified after creation, in contrast to an immutable object, whose state is fixed at the time of creation.

In LilyPond, mutable properties contain values that are specific to one grob. Typically, lists of other objects or results from computations are stored in mutable properties.

See also

Notation Reference: [immutable], page 774.

output-def

An instance of the Output-def class contains the methods and data structures associated with an output block. Instances are created for midi, layout and paper blocks.

parser

A parser analyzes the sequence of tokens produced by a lexer to determine its grammatical structure, grouping the tokens progressively into larger groupings according to the rules of the grammar. If the sequence of tokens is valid the end product is a tree of tokens whose root is the grammar’s start symbol. If this cannot be achieved the file is invalid and an appropriate
error message is produced. The syntactic groupings and the rules for constructing the groupings from their parts for the LilyPond syntax are defined in lily/parser.yy and shown in Backus Normal Form (BNF) in Section “LilyPond grammar” in Contributor’s Guide. This file is used to build the parser during the program build by the parser generator, Bison. It is part of the source code and is not included in the LilyPond binary installation.

**parser variable**

These are variables defined directly in Scheme. Their direct use by users is strongly discouraged, because their scoping semantics can be confusing.

When the value of such a variable is changed in a .ly file, the change is global, and unless explicitly reverted, the new value will persist to the end of the file, affecting subsequent \score blocks as well as external files added with the \include command. This can lead to unintended consequences and in complex typesetting projects the consequent errors can be difficult to track down.

LilyPond uses the following parser variables:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

**prob**

PRoperty OBjects, or probs for short, are instances of the Prob class, a simple base class for objects which have mutable and immutable property alists and the methods to manipulate them. The Music and Stream_event classes derive from Prob. Instances of the Prob class are also created to hold the formatted content of system grobs and titling blocks during page layout.

**smob**

Smobs, or ScheMe OBjects, are part of the mechanism used by Guile to export C and C++ objects to Scheme code. In LilyPond, smobs are created from C++ objects through macros. There are two types of smob objects: simple smobs, intended for simple immutable objects like numbers, and complex smobs, used for objects with identities. If you have access to the LilyPond sources, more information can be found in lily/includes/smob.hh.

**stencil**

An instance of the stencil class holds the information required to print a typographical object. It is a simple smob containing a confining box, which defines the vertical and horizontal extents of the object, and a Scheme expression which will print the object when evaluated. Stencils may be combined to form more complex stencils defined by a tree of Scheme expressions formed from the Scheme expressions of the component stencils.

The stencil property, which connects a grob to its stencil, is defined in the grob-interface interface.
See also
Internals Reference: Section “grob-interface” in Internals Reference.

A.17 All 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.

alternativeNumberingStyle (symbol)
The style of an alternative’s bar numbers. Can be numbers for going back to the
same number or numbers-with-letters for going back to the same number with
letter suffixes. No setting will not go back in measure-number time.

alternativeRestores (symbol list)
Timing variables that are restored to their value at the start of the first alternative
in subsequent alternatives.

associatedVoice (string)
Name of the context (see associatedVoiceType for its type, usually Voice) that
has the melody for this Lyrics line.

associatedVoiceType (symbol)
Type of the context that has the melody for this Lyrics line.

autoAccidentals (list)
List of different ways to typeset an accidental.
For determining when to print an accidental, several different rules are tried. The
rule that gives the highest number of accidentals is used.
Each entry in the list is either a symbol or a procedure.

symbol  The symbol is the name of the context in which the following rules are
to be applied. For example, if context is Section “Score” in Internals
Reference then all staves share accidentals, and if context is Section
“Staff” in Internals Reference then all voices in the same staff share
accidentals, but staves do not.

procedure The procedure represents an accidental rule to be applied to the previ-
ously specified context.
The procedure takes the following arguments:
context    The current context to which the rule should be applied.
Appendix A: Notation manual tables

pitch  The pitch of the note to be evaluated.
barnum  The current bar number.
measurepos  The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. (#t . #f) does not make sense.

autoBeamCheck (procedure)
A procedure taking three arguments, context, dir [start/stop (-1 or 1)], and test [shortest note in the beam]. A non-#f return value starts or stops the auto beam.

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

autoCautionaries (list)
List similar to autoAccidentals, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

automaticBars (boolean)
If set to false then bar lines will not be printed automatically; they must be explicitly created with a \bar command. Unlike the \cadenzaOn keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

barAlways (boolean)
If set to true a bar line is drawn after each note.

barCheckSynchronize (boolean)
If true then reset measurePosition when finding a bar check.

barNumberFormatter (procedure)
A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

barNumberVisibility (procedure)
A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the break-visibility property.

The following procedures are predefined:

all-bar-numbers-visible
Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

first-bar-number-invisible
Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn’t get a bar number either.

first-bar-number-invisible-save-broken-bars
Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

first-bar-number-invisible-and-no-parenthesized-bar-numbers
Enable bar numbers for all bars except the first bar and broken bars. This is the default.
(every-nth-bar-number-visible n)
    Assuming n is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

(modulo-bar-number-visible n m)
    If bar numbers 1, 4, 7, etc., should be enabled, n (the modulo) must be set to 3 and m (the division remainder) to 1.

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

beamExceptions (list)
    An alist of exceptions to autobeam rules that normally end on beats.

beamHalfMeasure (boolean)
    Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

beatStructure (list)
    List of baseMoments that are combined to make beats.

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. Values of 7 and -7 are common.

clefTranspositionFormatter (procedure)
    A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

clefTranspositionStyle (symbol)
    Determines the way the ClefModifier grob is displayed. Possible values are ‘default’, ‘parenthesized’ and ‘bracketed’.
completable (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 \( c2\times3 \), this specifies the scale factor to use for the newly-split notes and rests created by the engraver.
If \#f, the completion engraver uses the scale-factor of each duration being split.
If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

completionUnit (moment)
Sub-bar unit of completion.

connectArpeggios (boolean)
If set, connect arpeggios across piano staff.

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

createKeyOnClefChange (boolean)
Print a key signature whenever the clef is changed.

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

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

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

cueClefGlyph (string)
Name of the symbol within the music font.

cueClefPosition (number)
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

cueClefTransposition (integer)
Add this much extra transposition. Values of 7 and -7 are common.

cueClefTranspositionFormatter (procedure)
A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

cueClefTranspositionStyle (symbol)
Determines the way the ClefModifier grob is displayed. Possible values are ‘default’, ‘parenthesized’ and ‘bracketed’.

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

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

decrescendoText (markup)
The text to print at start of non-hairpin decrescendo, i.e., ‘dim.’.
**defaultBarType** (string)
Set the default type of bar line. See `whichBar` for information on available bar types. This variable is read by Section “Timing translator” in *Internals Reference* at Section “Score” in *Internals Reference* level.

**defaultStrings** (list)
A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

**doubleRepeatSegnoType** (string)
Set the default bar line for the combinations double repeat with segno. Default is `:'|S.|:'`.

**doubleRepeatType** (string)
Set the default bar line for double repeats.

**doubleSlurs** (boolean)
If set, two slurs are created for every slurred note, one above and one below the chord.

**drumPitchTable** (hash table)
A table mapping percussion instruments (symbols) to pitches.

**drumStyleTable** (hash table)
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

**endRepeatSegnoType** (string)
Set the default bar line for the combinations ending of repeat with segno. Default is `:'|S.`

**endRepeatType** (string)
Set the default bar line for the ending of repeats.

**explicitClefVisibility** (vector)
‘break-visibility’ function for clef changes.

**explicitCueClefVisibility** (vector)
‘break-visibility’ function for cue clef changes.

**explicitKeySignatureVisibility** (vector)
‘break-visibility’ function for explicit key changes. ‘\override’ of the break-visibility property will set the visibility for normal (i.e., at the start of the line) key signatures.

**extendersOverRests** (boolean)
Whether to continue extenders as they cross a rest.

**extraNatural** (boolean)
Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

**figuredBassAlterationDirection** (direction)
Where to put alterations relative to the main figure.
figuredBassCenterContinuations (boolean)
Whether to vertically center pairs of extender lines. This does not work with three
or more lines.

figuredBassFormatter (procedure)
A routine generating a markup for a bass figure.

figuredBassPlusDirection (direction)
Where to put plus signs relative to the main figure.

fingeringOrientations (list)
A list of symbols, containing ‘left’, ‘right’, ‘up’ and/or ‘down’. This list determines
where fingerings are put relative to the chord being fingered.

firstClef (boolean)
If true, create a new clef when starting a staff.

followVoice (boolean)
If set, note heads are tracked across staff switches by a thin line.

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

forbidBreak (boolean)
If set to #t, prevent a line break at this point.

forceClef (boolean)
Show clef symbol, even if it has not changed. Only active for the first clef after the
property is set, not for the full staff.

fretLabels (list)
A list of strings or Scheme-formatted markups containing, in the correct order, the
labels to be used for lettered frets in tablature.

glissandoMap (list)
A map in the form of '(((source1 . target1) (source2 . target2) (sourceN . targetN))
showing the glissandi to be drawn for note columns. The value '(' will default to
'((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two
note columns between which the glissandi occur.

gridInterval (moment)
Interval for which to generate 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.

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)
An alist that defines in what order alterations should be printed. The format is (step . alter), where step is a number from 0 to 6 and alter from -2 (sharp) to 2 (flat).

keyAlterations (list)
The current key signature. This is an alist containing (step . alter) or ((octave . step) . alter), where step is a number in the range 0 to 6 and alter a fraction, denoting alteration. For alterations, use symbols, e.g. keyAlterations = #`(6 . FLAT)).

lyricMelismaAlignment (number)
Alignment to use for a melisma syllable.

magnifyStaffValue (positive number)
The most recent value set with \magnifyStaff.

majorSevenSymbol (markup)
How should the major 7th be formatted in a chord name?
markFormatter (procedure)
A procedure taking as arguments the context and the rehearsal mark. It should
return the formatted mark as a markup object.

maximumFretStretch (number)
Don’t allocate frets further than this from specified frets.

measureLength (moment)
Length of one measure in the current time signature.

measurePosition (moment)
How much of the current measure have we had. This can be set manually to create
incomplete measures.

melismaBusyProperties (list)
A list of properties (symbols) to determine whether a melisma is playing. Setting
this property will influence how lyrics are aligned to notes. For example,
if set to 'melismaBusy beamMelismaBusy), only manual melismata and manual
beams are considered. Possible values include melismaBusy, slurMelismaBusy,
tieMelismaBusy, and beamMelismaBusy.

metronomeMarkFormatter (procedure)
How to produce a metronome markup. Called with two arguments: a
TempoChangeEvent and context.

middleCClefPosition (number)
The position of the middle C, as determined only by the clef. This can be calculated
by looking at clefPosition and clefGlyph.

middleCCuePosition (number)
The position of the middle C, as determined only by the clef of the cue notes. This
can be calculated by looking at cueClefPosition and cueClefGlyph.

middleCOffset (number)
The offset of middle C from the position given by middleCClefPosition This is
used for ottava brackets.

middleCPosition (number)
The place of the middle C, measured in half staff-spaces. Usually determined by
looking at middleCClefPosition and middleCOffset.

midiBalance (number)
Stereo balance for the MIDI channel associated with the current context. Ranges
from -1 to 1, where the values -1 (#LEFT), 0 (#CENTER) and 1 (#RIGHT) correspond
to leftmost emphasis, center balance, and rightmost emphasis, respectively.

midiChannelMapping (symbol)
How to map MIDI channels: per staff (default), instrument or voice.

midiChorusLevel (number)
Chorus effect level for the MIDI channel associated with the current context. Ranges
from 0 to 1 (0=off, 1=full effect).

midiExpression (number)
Expression control for the MIDI channel associated with the current context. Ranges
from 0 to 1 (0=off, 1=full effect).

midiInstrument (string)
Name of the MIDI instrument to use.
Appendix A: Notation manual tables

midiMaximumVolume (number)
   Analogous to midiMinimumVolume.

midiMergeUnisons (boolean)
   If true, output only one MIDI note-on event when notes with the same pitch, in the
   same MIDI-file track, overlap.

midiMinimumVolume (number)
   Set the minimum loudness for MIDI. Ranges from 0 to 1.

midiPanPosition (number)
   Pan position for the MIDI channel associated with the current context. Ranges
   from -1 to 1, where the values -1 (#LEFT), 0 (#CENTER) and 1 (#RIGHT) correspond
   to hard left, center, and hard right, respectively.

midiReverbLevel (number)
   Reverb effect level for the MIDI channel associated with the current context. Ranges
   from 0 to 1 (0=off, 1=full effect).

minimumFret (number)
   The tablature auto string-selecting mechanism selects the highest string with a fret
   at least minimumFret.

minimumPageTurnLength (moment)
   Minimum length of a rest for a page turn to be allowed.

minimumRepeatLengthForPageTurn (moment)
   Minimum length of a repeated section for a page turn to be allowed within that
   section.

minorChordModifier (markup)
   Markup displayed following the root for a minor chord

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

noteNameFunction (procedure)
   Function used to convert pitches into strings and markups.

noteNameSeparator (string)
   String used to separate simultaneous NoteName objects.

noteToFretFunction (procedure)
   Convert list of notes and list of defined strings to full list of strings and fret numbers.
   Parameters: The context, a list of note events, a list of tabstring events, and the
   fretboard grob if a fretboard is desired.

nullAccidentals (boolean)
   The Accidental_engraver generates no accidentals for notes in contexts were this
   is set. In addition to supressing the printed accidental, this option removes any
   effect the note would have had on accidentals in other voices.

ottavation (markup)
   If set, the text for an ottava spanner. Changing this creates a new text spanner.

ottavationMarkups (list)
   An alist defining the markups used for ottava brackets. It contains entries of the
   form (number of octaves . markup).

output (music output)
   The output produced by a score-level translator during music interpretation.
partCombineForced (symbol)
    Override for the partCombine decision. Can be apart, chords, unisono, solo1, or solo2.

partCombineTextsOnNote (boolean)
    Print part-combine texts only on the next note rather than immediately on rests or skips.

pedalSostenutoStrings (list)
    See pedalSustainStrings.

pedalSostenutoStyle (symbol)
    See pedalSustainStyle.

pedalSustainStrings (list)
    A list of strings to print for sustain-pedal. Format is (up updown down), where each of the three is the string to print when this is done with the pedal.

pedalSustainStyle (symbol)
    A symbol that indicates how to print sustain pedals: text, bracket or mixed (both).

pedalUnaCordaStrings (list)
    See pedalSustainStrings.

pedalUnaCordaStyle (symbol)
    See pedalSustainStyle.

predefinedDiagramTable (hash table)
    The hash table of predefined fret diagrams to use in FretBoards.

printAccidentalNames (boolean or symbol)
    Print accidentals in the NoteNames context.

printKeyCancellation (boolean)
    Print restoration alterations before a key signature change.

printNotesLanguage (string)
    Use a specific language in the NoteNames context.

printOctaveNames (boolean or symbol)
    Print octave marks in the NoteNames context.

printPartCombineTexts (boolean)
    Set ‘Solo’ and ‘A due’ texts in the part combiner?

proportionalNotationDuration (moment)
    Global override for shortest-playing duration. This is used for switching on proportional notation.

rehearsalMark (integer)
    The last rehearsal mark printed.

repeatCommands (list)
    This property is a list of commands of the form (list 'volta x), where x is a string or #f. 'end-repeat is also accepted as a command.

repeatCountVisibility (procedure)
    A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when countPercentRepeats is set.

restCompletionBusy (boolean)
    Signal whether a completion-rest is active.
Appendix A: Notation manual tables

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

restrainOpenStrings (boolean)
Exclude open strings from the automatic fret calculator.

searchForVoice (boolean)
Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

segnoType (string)
Set the default bar line for a requested segno. Default is ‘S’.

shapeNoteStyles (vector)
Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

shortInstrumentName (markup)
See instrumentName.

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

skipBars (boolean)
If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

\set Score.skipBars= ##t
r1 r1*3 R1*3
{/}

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

slashChordSeparator (markup)
The markup object used to separate a chord name from its root note in case of inversions or slash chords.

soloIIText (markup)
The text for the start of a solo for voice ‘two’ when part-combining.

soloText (markup)
The text for the start of a solo when part-combining.

squashedPosition (integer)
Vertical position of squashing for Section “Pitch_squash_engraver” in Internals Reference.

staffLineLayoutFunction (procedure)
Layout of staff lines, traditional, or semitone.

stanza (markup)
Stanza ‘number’ to print before the start of a verse. Use in Lyrics context.

startAtNoteColumn (boolean)
Start DurationLine grob at entire NoteColumn.
Appendix A: Notation manual tables

startAtSkip (boolean)
Start DurationLine grob at skip-event.

startRepeatSegnoType (string)
Set the default bar line for the combinations beginning of repeat with segno. Default is ‘S.|:’.

startRepeatType (string)
Set the default bar line for the beginning of repeats.

stemLeftBeamCount (integer)
Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

stemRightBeamCount (integer)
See stemLeftBeamCount.

strictBeatBeaming (boolean)
Should partial beams reflect the beat structure even if it causes flags to hang out?

stringNumberOrientations (list)
See fingeringOrientations.

stringOneTopmost (boolean)
Whether the first string is printed on the top line of the tablature.

stringTunings (list)
The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

strokeFingerOrientations (list)
See fingeringOrientations.

subdivideBeams (boolean)
If set, multiple beams will be subdivided at baseMoment positions by only drawing one beam over the beat.

suggestAccidentals (boolean 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 $2^\frac{1}{7}$.

suspendRestMerging (boolean)
When using the Merge_rest_engraver do not merge rests when this is set to true.

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

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

tablatureFormat (procedure)
A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

tabStaffLineLayoutFunction (procedure)
A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.
Appendix A: Notation manual tables

[tempoHideNote (boolean)]
    Hide the note = count in tempo marks.

tempoWholesPerMinute (moment)
    The tempo in whole notes per minute.

tieWaitForNote (boolean)
    If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

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

timeSignatureSettings (list)
    A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for baseMoment, beatStructure, and beamExceptions.

timing (boolean)
    Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

tonics (pitch)
    The tonic of the current scale.

topLevelAlignment (boolean)
    If true, the Vertical_align_ engraver will create a VerticalAlignment; otherwise, it will create a StaffGrouper.

tupletFullLength (boolean)
    If set, the tuplet is printed up to the start of the next note.

tupletFullLengthNote (boolean)
    If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

tupletSpannerDuration (moment)
    Normally, a tuplet bracket is as wide as the \times expression that gave rise to it. By setting this property, you can make brackets last shorter.

    
    { 
    \set tupletSpannerDuration = #(ly:make-moment 1 4) 
    \times 2/3 { c8 c c c c c } 
    } 

useBassFigureExtenders (boolean)
    Whether to use extender lines for repeated bass figures.

vocalName (markup)
    Name of a vocal line.

voltaSpannerDuration (moment)
    This specifies the maximum duration to use for the brackets printed for \alternative. This can be used to shrink the length of brackets in the situation where one alternative is very large.

whichBar (string)
    This property is read to determine what type of bar line to create.

    Example:

    \set Staff.whichBar = ".|:"
This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

### A.18 Layout properties

**add-stem-support** (boolean)
- If set, the Stem object is included in this script’s support.

**after-line-breaking** (boolean)
- Dummy property, used to trigger callback for **after-line-breaking**.

**align-dir** (direction)
- Which side to align? -1: left side, 0: around center of width, 1: right side.

**allow-loose-spacing** (boolean)
- If set, column can be detached from main spacing.

**allow-span-bar** (boolean)
- If false, no inter-staff bar line will be created below this bar line.

**alteration** (number)
- Alteration numbers for accidental.

**alteration-alist** (list)
- List of (pitch , accidental) pairs for key signature.

**annotation-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 beam-
let is determined by taking either the default length or the length specified by
beamlet-max-length-proportion, whichever is smaller.

beamlet-max-length-proportion (pair)
The maximum length of a beamlet, as a proportion of the distance between two
adjacent stems.

before-line-breaking (boolean)
Dummy property, used to trigger a callback function.

between-cols (pair)
Where to attach a loose column to.

bound-details (list)
An alist of properties for determining attachments of spanners to edges.

bound-padding (number)
The amount of padding to insert around spanner bounds.

bracket-flare (pair of numbers)
A pair of numbers specifying how much edges of brackets should slant outward.
Value 0.0 means straight edges.
Appendix A: Notation manual tables

**bracket-visibility** (boolean or symbol)
This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to **if-no-beam** makes it print only if there is no beam associated with this tuplet bracket.

**break-align-anchor** (number)
Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

**break-align-anchor-alignment** (number)
Read by **ly:break-aligned-interface::calc-extent-aligned-anchor** for aligning an anchor to a grob’s extent.

**break-align-orders** (vector)
This is a vector of 3 lists: #(end-of-line unbroken start-of-line). Each list contains **break-align symbols** that specify an order of breakable items (see Section “break-alignment-interface” in Internals Reference). For example, this places time signatures before clefs:

```latex
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
cue-end-clef
ambitus
breathing-sign
time-signature
clef
cue-clef
staff-bar
key-cancellation
key-signature
custos))
```

**break-align-symbol** (symbol)
This key is used for aligning, ordering, and spacing breakable items. See Section “break-alignment-interface” in Internals Reference.

**break-align-symbols** (list)
A list of **break-align symbols** that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to **break-visibility**, we will align to the next grob (and so on). Choices are listed in Section “break-alignment-interface” in Internals Reference.

**break-overshoot** (pair of numbers)
How much does a broken spanner stick out of its bounds?

**break-visibility** (vector)
A vector of 3 booleans, #(end-of-line unbroken begin-of-line). #t means visible, #f means killed.

**breakable** (boolean)
Allow breaks here.

**broken-bound-padding** (number)
The amount of padding to insert when a spanner is broken at a line break.

**chord-dots-limit** (integer)
Limits the column of dots on each chord to the height of the chord plus **chord-dots-limit** staff-positions.
circled-tip (boolean)
Put a circle at start/end of hairpins (al/del niente).

clef-alignments (list)
An alist of parent-alignments that should be used for clef modifiers with various clefs

clip-edges (boolean)
Allow outward pointing beamlets at the edges of beams?

collapse-height (dimension, in staff space)
Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

collision-interfaces (list)
A list of interfaces for which automatic beam-collision resolution is run.

collision-voice-only (boolean)
Does automatic beam 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.
Appendix A: Notation manual tables

**default-direction** (direction)
Direction determined by note head positions.

**default-staff-staff-spacing** (list)
The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

**details** (list)
A list of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

**digit-names** (vector)
Names for string finger digits.

**direction** (direction)
If `side-axis` is 0 (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP=1`, `DOWN=-1`, `LEFT=-1`, `RIGHT=1`, `CENTER=0`.

**dot-count** (integer)
The number of dots.

**dot-negative-kern** (number)
The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

**dot-placement-list** (list)
List consisting of `(description string-number fret-number finger-number)` entries used to define fret diagrams.

**double-stem-separation** (number)
The distance between the two stems of a half note in tablature when using `{\tabFullNotation}`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

**duration-log** (integer)
The 2-log of the note head duration, i.e., 0 = whole note, 1 = half note, etc.

**eccentricity** (number)
How asymmetrical to make a slur. Positive means move the center to the right.

**edge-height** (pair)
A pair of numbers specifying the heights of the vertical edges: `(left-height . right-height)`.

**edge-text** (pair)
A pair specifying the texts to be set at the edges: `(left-text . right-text)`.

**expand-limit** (integer)
Maximum number of measures expanded in church rests.

**extra-dy** (number)
Slope glissandi this much extra.

**extra-offset** (pair of numbers)
A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff’s `StaffSymbol`. 
extra-spacing-height (pair of numbers)
In the horizontal spacing problem, we increase the height of each item by this amount
(by adding the ‘car’ to the bottom of the item and adding the ‘cdr’ to the top of
the item). In order to make a grob infinitely high (to prevent the horizontal spacing
problem from placing any other grobs above or below this grob), set this to \((-\inf.0 . +\inf.0)\).

extra-spacing-width (pair of numbers)
In the horizontal spacing problem, we pad each item by this amount (by adding the
‘car’ on the left side of the item and adding the ‘cdr’ on the right side of the item).
In order to make a grob take up no horizontal space at all, set this to \((+\inf.0 . -\inf.0)\).

flag-count (number)
The number of tremolo beams.

flag-style (symbol)
The style of the flag to be used with MetronomeMark. Available are
'modern-straight-flag, 'old-straight-flag, flat-flag, mensural and
'default

flat-positions (list)
Flats in key signatures are placed within the specified ranges of staff-positions. The
general form is a list of pairs, with one pair for each type of clef, in order of the
staff-position at which each clef places C: (alto treble tenor soprano baritone
mezzosoprano bass). If the list contains a single element it applies for all clefs. A
single number in place of a pair sets accidentals within the octave ending at that
staff-position.

font-encoding (symbol)
The font encoding is the broadest category for selecting a font. Currently, only
lilypond’s system fonts (Emmentaler) are using this property. Available values are
fetaMusic (Emmentaler), fetaBraces, fetaText (Emmentaler).

font-family (symbol)
The font family is the broadest category for selecting text fonts. Options include:
sans, roman.

font-features (list)
Opentype features.

font-name (string)
Specifies a file name (without extension) of the font to load. This setting overrides
selection using font-family, font-series and font-shape.

font-series (symbol)
Select the series of a font. Choices include medium, bold, bold-narrow, etc.

font-shape (symbol)
Select the shape of a font. Choices include upright, italic, caps.

font-size (number)
The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, ~1 is
smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly
a factor 2 larger. If the context property fontSize is set, its value is added to this
before the glyph is printed. Fractional values are allowed.

footnote (boolean)
Should this be a footnote or in-note?
footnote-music (music)
Music creating a footnote.

footnote-text (markup)
A footnote for the grob.

force-hshift (number)
This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Section “note-collision-interface” in Internals Reference.

forced-spacing (number)
Spacing forced between grobs, used in various ligature engravers.

fraction (fraction, as pair)
Numerator and denominator of a time signature object.

defrench-beaming (boolean)
Use French beaming style for this stem. The stem stops at the innermost beams.

fret-diagram-details (list)
An alist of detailed grob properties for fret diagrams. Each alist entry consists of a (property . value) pair. The properties which can be included in fret-diagram-details include the following:

- barre-type – Type of barre indication used. Choices include curved, straight, and none. Default curved.
- capo-thickness – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
- dot-color – Color of dots. Options include black and white. Default black.
- dot-label-font-mag – Magnification for font used to label fret dots. Default value 1.
- dot-position – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-dot-radius for dots with labels.
- dot-radius – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
- finger-code – Code for the type of fingering indication used. Options include none, in-dot, and below-string. Default none for markup fret diagrams, below-string for FretBoards fret diagrams.
- fret-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 roman-lower, roman-upper, arabic and custom. In the later case, the format string is supplied by the fret-label-custom-format property. Default roman-lower.

• **open-string** – Character string to be used to indicate open string. Default "o".

• **orientation** – Orientation of fret-diagram. Options include normal, landscape, and opposing-landscape. Default normal.

• **string-count** – The number of strings. Default 6.

• **string-distance** – Multiplier to adjust the distance between strings. Default 1.0.

• **string-label-font-mag** – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for normal orientation, 0.5 for landscape and opposing-landscape.

• **string-thickness-factor** – Factor for changing thickness of each string in the fret diagram. Thickness of string $k$ is given by \( \text{thickness} \times (1 + \text{string-thickness-factor})^{(k-1)} \). 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-name (string)
The glyph name within the font.
In the context of (span) bar lines, glyph-name represents a processed form of glyph, where decisions about line breaking etc. are already taken.

glyph-name-alist (list)
An alist of key-string pairs.

graphical (boolean)
Display in graphical (vs. text) form.

grow-direction (direction)
Crescendo or decrescendo?

hair-thickness (number)
Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is not influenced by changes to Staff.StaffSymbol.thickness).

harp-pedal-details (list)
An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a (property . value) pair. The properties which can be included in harp-pedal-details include the following:

- box-offset – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
- box-width – Width of each pedal box. Default value 0.4.
- box-height – Height of each pedal box. Default value 1.0.
- space-before-divider – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
- space-after-divider – Space between boxes after the first divider. Default value 0.8.
- circle-thickness – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
- circle-x-padding – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
- circle-y-padding – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

head-direction (direction)
Are the note heads left or right in a semitie?

height (dimension, in staff space)
Height of an object in staff-space units.

height-limit (dimension, in staff space)
Maximum slur height: The longer the slur, the closer it is to this height.

hide-tied-accidental-after-break (boolean)
If set, an accidental that appears on a tied note after a line break will not be displayed.

horizon-padding (number)
The amount to pad the axis along which a Skyline is built for the side-position-interface.
horizontal-shift (integer)
An integer that identifies ranking of 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 (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 (list)
   An alist of properties for determining attachments of spanners to edges.

left-padding (dimension, in staff space)
   The amount of space that is put left to an object (e.g., a lyric extender).

length (dimension, in staff space)
   User override for the stem length of unbeamed stems (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 (list)
   An alist of properties to use if this column is the start of a system.

line-count (integer)
   The number of staff lines.

line-positions (list)
   Vertical positions of staff lines.

line-thickness (number)
   For slurs and ties, this is the diameter of the virtual “pen” that draws the two arcs
   of the curve’s outline, which intersect at the endpoints. This property is expressed
   as a multiple of the current staff-line thickness (i.e. the visual output is influenced
   by changes to Staff.StaffSymbol.thickness).

long-text (markup)
   Text markup. See Section “Formatting text” in Notation Reference.

max-beam-connect (integer)
   Maximum number of beams to connect to beams from this stem. Further beams
   are typeset as beamlets.

max-symbol-separation (number)
   The maximum distance between symbols making up a church rest.

maximum-gap (number)
   Maximum value allowed for gap property.

measure-count (integer)
   The number of measures for a multi-measure rest.

measure-length (moment)
   Length of a measure. Used in some spacing situations.
Appendix A: Notation manual tables

merge-differently-dotted (boolean)
  Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

merge-differently-dotted only applies to opposing stem directions (i.e., voice 1 & 2).

merge-differently-headed (boolean)
  Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation.
  The value of this setting is used by Section “note-collision-interface” in Internals Reference.

merge-differently-headed only applies to opposing stem directions (i.e., voice 1 & 2).

minimum-distance (dimension, in staff space)
  Minimum distance between rest and notes or beam.

minimum-length (dimension, in staff space)
  Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the springs-and-rods property. If added to a Tie, this sets the minimum distance between noteheads.

minimum-length-after-break (dimension, in staff space)
  If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the springs-and-rods property. If added to a Tie, this sets the minimum distance to the notehead.

minimum-length-fraction (number)
  Minimum length of ledger line as fraction of note head size.

minimum-space (dimension, in staff space)
  Minimum distance that the victim should move (after padding).

minimum-X-extent (pair of numbers)
  Minimum size of an object in X dimension, measured in staff-space units.

minimum-Y-extent (pair of numbers)
  Minimum size of an object in Y dimension, measured in staff-space units.

neutral-direction (direction)
  Which direction to take in the center of the staff.

neutral-position (number)
  Position (in half staff spaces) where to flip the direction of custos stem.

next (graphical (layout) object)
  Object that is next relation (e.g., the lyric syllable following an extender).

no-alignment (boolean)
  If set, don’t place this grob in a VerticalAlignment; rather, place it using its own Y-offset callback.

no-ledgers (boolean)
  If set, don’t draw ledger lines on this object.

no-stem-extend (boolean)
  If set, notes with ledger lines do not get stems extending to the middle staff line.

non-break-align-symbols (list)
  A list of symbols that determine which NON-break-aligned interfaces to align this to.
non-default (boolean)
Set for manually specified clefs and keys.

non-musical (boolean)
True if the grob belongs to a NonMusicalPaperColumn.

nonstaff-nonstaff-spacing (list)
The spacing alist controlling the distance between the current non-staff line and
the next non-staff line in the direction of staff-affinity, if both are on the
same side of the related staff, and staff-affinity is either UP or DOWN. See
staff-staff-spacing for a description of the alist structure.

nonstaff-relatedstaff-spacing (list)
The spacing alist controlling the distance between the current non-staff line and
the nearest staff in the direction of staff-affinity, if there are no non-staff lines
between the two, and staff-affinity is either UP or DOWN. If staff-affinity is
CENTER, then nonstaff-relatedstaff-spacing is used for the nearest staves on
both sides, even if other non-staff lines appear between the current one and either
of the staves. See staff-staff-spacing for a description of the alist structure.

nonstaff-unrelatedstaff-spacing (list)
The spacing alist controlling the distance between the current non-staff line and
the nearest staff in the opposite direction from staff-affinity, if there are no other
non-staff lines between the two, and staff-affinity is either UP or DOWN. See
staff-staff-spacing for a description of the alist structure.

normalized-endpoints (pair)
Represents left and right placement over the total spanner, where the width of the
spanner is normalized between 0 and 1.

note-collision-threshold (dimension, in staff space)
Simultaneous notes that are this close or closer in units of staff-space will be
identified as vertically colliding. Used by Stem grobs for notes in the same voice,
and NoteCollision grobs for notes in different voices. Default value 1.

note-names (vector)
Vector of strings containing names for easy-notation note heads.

number-type (symbol)
Numbering style. Choices include roman-lower, roman-upper and arabic.

output-attributes (list)
An alist of attributes for the grob, to be included in output files. When the
SVG typesetting backend is used, the attributes are assigned to a group (<g>)
containing all of the stencils that comprise a given grob. For example, '(((id .
123) (class . foo) (data-whatever . \bar")) will produce <g id="123"
class="foo" data-whatever="\bar"> ... </g>. In the Postscript backend, where
there is no way to group items, the setting of the output-attributes property will
have no effect.

outside-staff-horizontal-padding (number)
By default, an outside-staff-object can be placed so that is it very close to another
grob horizontally. If this property is set, the outside-staff-object is raised so that it
is not so close to its neighbor.

outside-staff-padding (number)
The padding to place between grobs when spacing according to
outside-staff-priority. Two grobs with different outside-staff-padding
values have the larger value of padding between them.
outside-staff-placement-directive (symbol)
One of four directives telling how outside staff objects should be placed.
- left-to-right-greedy – Place each successive grob from left to right.
- left-to-right-polite – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
- right-to-left-greedy – Same as left-to-right-greedy, but from right to left.
- right-to-left-polite – Same as left-to-right-polite, but from right to left.

outside-staff-priority (number)
If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller outside-staff-priority is closer to the staff.

packed-spacing (boolean)
If set, the notes are spaced as tightly as possible.

padding (dimension, in staff space)
Add this much extra space between objects that are next to each other.

padding-pairs (list)
An alist mapping (name . name) to distances.

page-break-penalty (number)
Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

page-break-permission (symbol)
Instructs the page breaker on whether to put a page break at this column. Can be force or allow.

page-number (number)
Page number on which this system ends up.

page-turn-penalty (number)
Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

page-turn-permission (symbol)
Instructs the page breaker on whether to put a page turn at this column. Can be force or allow.

parent-alignment-X (number)
Specify on which point of the parent the object is aligned. The value -1 means aligned on parent’s left edge, 0 on center, and 1 right edge, in X direction. Other numerical values may also be specified - the unit is half the parent’s width. If unset, the value from self-alignment-X property will be used.

parent-alignment-Y (number)
Like parent-alignment-X but for the Y axis.

parenthesis-friends (list)
A list of Grob types, as symbols. When parentheses enclose a Grob that has ’parenthesis-friends, the parentheses widen to include any child Grobs with type among ’parenthesis-friends.
parenthesized (boolean)
Parenthesize this grob.

positions (pair of numbers)
Pair of staff coordinates (left, right), where both left and right are in staff-space units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

prefer-dotted-right (boolean)
For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

protrusion (number)
In an arpeggio bracket, the length of the horizontal edges.

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 (list)
Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

restore-first (boolean)
Print a natural before the accidental.

rhythmic-location (rhythmic location)
Where (bar number, measure position) in the score.

right-bound-info (list)
An alist of properties for determining attachments of spanners to edges.

right-padding (dimension, in staff space)
Space to insert on the right side of an object (e.g., between note and its accidentals).

rotation (list)
Number of degrees to rotate this object, and what point to rotate around. For example, '(45 0 0) rotates by 45 degrees around the center of this object.

round-up-exceptions (list)
A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See round-up-to-longer-rest.
round-up-to-longer-rest (boolean)
Displays the longer multi-measure rest when the length of a measure is between two values of usable-duration-logs. For example, displays a breve instead of a whole in a 3/2 measure.

rounded (boolean)
Decide whether lines should be drawn rounded or not.

same-direction-correction (number)
Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

script-priority (number)
A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

segno-kern (number)
The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is not influenced by changes to Staff.StaffSymbol.thickness).

self-alignment-X (number)
Specify alignment of an object. The value ~1 means left aligned, 0 centered, and 1 right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

self-alignment-Y (number)
Like self-alignment-X but for the Y axis.

shape (symbol)
This setting determines what shape a grob has. Valid choices depend on the stencil callback reading this property.

sharp-positions (list)
Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: (alto treble tenor soprano baritone mezzosoprano bass). If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

shorten-pair (pair of numbers)
The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

shortest-duration-space (number)
Start with this multiple of spacing-increment space for the shortest duration. See also Section “spacing-spanner-interface” in Internals Reference.

shortest-playing-duration (moment)
The duration of the shortest note playing here.

shortest-starter-duration (moment)
The duration of the shortest note that starts here.
**side-axis** (number)
   If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

**side-relative-direction** (direction)
   Multiply direction of direction-source with this to get the direction of this object.

**simple-Y** (boolean)
   Should the Y placement of a spanner disregard changes in system heights?

**size** (number)
   The ratio of the size of the object to its default size.

**skip-quanting** (boolean)
   Should beam quanting be skipped?

**skyline-horizontal-padding** (number)
   For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

**skyline-vertical-padding** (number)
   The amount by which the left and right skylines of a column are padded vertically, beyond the Y-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** (list)
   An alist that specifies distances from this grob to other breakable items, using the format:

   `(('break-align-symbol . (spacing-style . space))
    (break-align-symbol . (spacing-style . space))
     ...)

   Standard choices for break-align-symbol are listed in Section “break-alignment-interface” in Internals Reference. Additionally, three special break-align symbols available to space-alist are:

   **first-note**
      used when the grob is just left of the first note on a line

   **next-note**
      used when the grob is just left of any other note; if not set, the value of first-note gets used
right-edge
used when the grob is the last item on the line (only compatible with the extra-space spacing style)

Choices for spacing-style are:

extra-space
Put this much space between the two grobs. The space is stretchable when paired with first-note or next-note; otherwise it is fixed.

minimum-space
Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with first-note or next-note; otherwise it is fixed. Not compatible with right-edge.

fixed-space
Only compatible with first-note and next-note. Put this much fixed space between the grob and the note.

minimum-fixed-space
Only compatible with first-note and next-note. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

semi-fixed-space
Only compatible with first-note and next-note. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

space-to-barline (boolean)
If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

spacing-increment (dimension, in staff space)
The unit of length for note-spacing. Typically, the width of a note head. See also Section “spacing-spanner-interface” in Internals Reference.

spacing-pair (pair)
A pair of alignment symbols which set an object’s spacing relative to its left and right BreakAlignments.

For example, a MultiMeasureRest will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

\override MultiMeasureRest.spacing-pair = #'(staff-bar . staff-bar)

spanner-id (index or symbol)
An identifier to distinguish concurrent spanners.

springs-and-rods (boolean)
Dummy variable for triggering spacing routines.
stacking-dir (direction)
  Stack objects in which direction?

staff-affinity (direction)
  The direction of the staff to use for spacing the current non-staff line. Choices are UP, DOWN, and CENTER. If CENTER, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting staff-affinity for a staff causes it to be treated as a non-staff line. Setting staff-affinity to #f causes a non-staff line to be treated as a staff.

staff-padding (dimension, in staff space)
  Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics p and f) on their baselines.

staff-position (number)
  Vertical position, measured in half staff spaces, counted from the middle line.

staff-space (dimension, in staff space)
  Amount of space between staff lines, expressed in global staff-space.

staff-staff-spacing (list)
  When applied to a staff-group’s StaffGrouper grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff’s VerticalAxisGroup grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the StaffGrouper grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:
  - basic-distance – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - minimum-distance – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
  - padding – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
  - stretchability – a unitless measure of the dimension’s relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

staffgroup-staff-spacing (list)
  The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the staff-staff-spacing property of the staff’s VerticalAxisGroup grob is set, that is used instead. See staff-staff-spacing for a description of the alist structure.

stem-attachment (pair of numbers)
  An (x . y) pair where the stem attaches to the notehead.

stem-begin-position (number)
  User override for the begin position of a stem.
stem-spacing-correction (number)
   Optical correction amount for stems that are placed in tight configurations. For
   opposite directions, this amount is the correction for two normal sized stems that
   overlap completely.

stemlet-length (number)
   How long should be a stem over a rest?

stencil (stencil)
   The symbol to print.

stencils (list)
   Multiple stencils, used as intermediate value.

strict-grace-spacing (boolean)
   If set, main notes are spaced normally, then grace notes are put left of the musical
   columns for the main notes.

strict-note-spacing (boolean)
   If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not
   spaced separately, but put before musical columns.

stroke-style (string)
   Set to "grace" to turn stroke through flag on.

style (symbol)
   This setting determines in what style a grob is typeset. Valid choices depend on the
   stencil callback reading this property.

text (markup)
   Text markup. See Section “Formatting text” in Notation Reference.

text-direction (direction)
   This controls the ordering of the words. The default RIGHT is for roman text. Arabic
   or Hebrew should use LEFT.

thick-thickness (number)
   Thickness of the thick line in a bar line, expressed as a multiple of the de-
   fault staff-line thickness (i.e. the visual output is not influenced by changes to
   Staff.StaffSymbol.thickness).

thickness (number)
   For grobs made up of lines, this is the thickness of the line. For slurs and ties, this
   is the distance between the two arcs of the curve’s outline at its thickest point, not
   counting the diameter of the virtual “pen” that draws the arcs. This property is
   expressed as a multiple of the current staff-line thickness (i.e. the visual output is
   influenced by changes to Staff.StaffSymbol.thickness).

tie-configuration (list)
   List of (position, dir) pairs, indicating the desired tie configuration, where posi-
   tion is the offset from the center of the staff in staff space and dir indicates the
   direction of the tie (1=>up, -1=>down, 0=>center). A non-pair entry in the list
   causes the corresponding tie to be formatted automatically.

to-barline (boolean)
   If true, the spanner will stop at the bar line just before it would otherwise stop.

toward-stem-shift (number)
   Amount by which scripts are shifted toward the stem if their direction coincides
   with the stem direction. 0.0 means centered on the note head (the default position
   of most scripts); 1.0 means centered on the stem. Interpolated values are possible.
Appendix A: Notation manual tables

\textbf{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 \texttt{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.

\textbf{transparent} (boolean)
This makes the grob invisible.

\textbf{tuplet-slur} (boolean)
Draw a slur instead of a bracket for tuplets.

\textbf{uniform-stretching} (boolean)
If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

\textbf{usable-duration-logs} (list)
List of \texttt{duration-logs} that can be used in typesetting the grob.

\textbf{use-skylines} (boolean)
Should skylines be used for side positioning?

\textbf{used} (boolean)
If set, this spacing column is kept in the spacing problem.

\textbf{vertical-skylines} (pair of skylines)
Two skylines, one above and one below this grob.

\textbf{voiced-position} (number)
The staff-position of a voiced \texttt{Rest}, negative if the rest has direction \texttt{DOWN}.

\textbf{when} (moment)
Global time step associated with this column.

\textbf{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 staffline thickness. The \texttt{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 \texttt{line-thickness}. The shape of the background is determined by \texttt{whiteout-style}. Usually \texttt{#f} by default.

\textbf{whiteout-style} (symbol)
Determines the shape of the \texttt{whiteout} background. Available are \texttt{outline}, \texttt{rounded-box}, and the default \texttt{box}. There is one exception: Use \texttt{special} for \texttt{LyricHyphen}.

\textbf{width} (dimension, in staff space)
The width of a grob measured in staff space.

\textbf{word-space} (dimension, in staff space)
Space to insert between words in texts.

\textbf{X-align-on-main-noteheads} (boolean)
If true, this grob will ignore suspended noteheads when aligning itself on \texttt{NoteColumn}.

\textbf{X-extent} (pair of numbers)
Extent (size) in the X direction, measured in staff-space units, relative to object’s reference point.
X-offset (number)
The horizontal amount that this object is moved relative to its X-parent.

X-positions (pair of numbers)
Pair of X staff coordinates of a spanner in the form \texttt{(left . right)}, where both \texttt{left} and \texttt{right} are in staff-space units of the current staff.

Y-extent (pair of numbers)
Extent (size) in the Y direction, measured in staff-space units, relative to object’s reference point.

Y-offset (number)
The vertical amount that this object is moved relative to its Y-parent.

zigzag-length (dimension, in staff space)
The length of the lines of a zigzag, relative to zigzag-width. A value of 1 gives 60-degree zigzags.

zigzag-width (dimension, in staff space)
The width of one zigzag squiggle. This number is adjusted slightly so that the spanner line can be constructed from a whole number of squiggles.

A.19 Available music functions

\texttt{\textbackslash absolute} [music] - music (music)
Make music absolute. This does not actually change the music itself but rather hides it from surrounding \texttt{\textbackslash relative} and \texttt{\textbackslash fixed} commands.

\texttt{\textbackslash acciaccatura} [music] - music (music)
Create an acciaccatura from the following music expression

\texttt{\textbackslash accidentalStyle} [music] - style (symbol list)
Set accidental style to symbol list \texttt{style} in the form \texttt{`piano-cautionary’}. If \texttt{style} has a form like \texttt{`Staff.piano-cautionary’}, the settings are applied to that context. Otherwise, the context defaults to \texttt{`Staff’}, except for piano styles, which use \texttt{`GrandStaff’} as a context.

\texttt{\textbackslash addChordShape} [void] - key-symbol (symbol) tuning (pair) shape-definition (string or pair)
Add chord shape \texttt{shape-definition} to the chord-shape-table hash with the key \texttt{(cons key-symbol tuning)}.

\texttt{\textbackslash addInstrumentDefinition} [void] - name (string) lst (list)
Create instrument \texttt{name} with properties \texttt{list}.

\texttt{\textbackslash addQuote} [void] - name (string) music (music)
Define music as a quotable music expression named \texttt{name}

\texttt{\textbackslash afterGrace} [music] - fraction [non-negative rational, fraction, or moment] main (music) grace (music)
Create grace note(s) after a \texttt{main} music expression.
The musical position of the grace expression is after a given fraction of the main note’s duration has passed. If \texttt{fraction} is not specified as first argument, it is taken from afterGraceFraction which has a default value of \texttt{3/4}.

\texttt{\textbackslash allowPageTurn} [music]
Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

\texttt{\textbackslash allowVoltaHook} [void] - bar (string)
Allow the volta bracket hook being drawn over bar line \texttt{bar}.
\texttt{\texttt{\textbackslash alterBroken}} [music] - property (key list or symbol) arg (list) item (key list or music)

Override property for pieces of broken spanner item with values arg. item may either be music in the form of a starting spanner event, or a symbol list in the form ‘\texttt{\texttt{Context.Grob}}’ or just ‘\texttt{\texttt{Grob}}’. If item is in the form of a spanner event, property may also have the form ‘\texttt{\texttt{Grob.property}}’ for specifying a directed tweak.

\texttt{\texttt{\textbackslash ambitusAfter}} [music] - target (symbol)

Move the ambitus after the break-align symbol target.

\texttt{\texttt{\textbackslash appendToTag}} [music] - tag (symbol) more (music) music (music)

Append more to the elements of all music expressions in music that are tagged with tag.

\texttt{\texttt{\textbackslash applyContext}} [music] - proc (procedure)

Modify context properties with Scheme procedure proc.

\texttt{\texttt{\textbackslash applyMusic}} [music] - func (procedure) music (music)

Apply procedure func to music.

\texttt{\texttt{\textbackslash applyOutput}} [music] - target (symbol list or symbol) proc (procedure)

Apply function proc to every layout object matched by target which takes the form Context or Context.Grob.

\texttt{\texttt{\textbackslash appoggiatura}} [music] - music (music)

Create an appoggiatura from music

\texttt{\texttt{\textbackslash assertBeamQuant}} [music] - l (pair) r (pair)

Testing function: check whether the beam quants l and r are correct

\texttt{\texttt{\textbackslash assertBeamSlope}} [music] - comp (procedure)

Testing function: check whether the slope of the beam is the same as comp


Make voices that switch between staves automatically. As an option the pitch where to switch staves may be specified. The clefs for the staves are optional as well. Setting clefs works only for implicitly instantiated staves.

\texttt{\texttt{\textbackslash balloonGrobText}} [music] - grob-name (symbol) offset (pair of numbers) text (markup)

Attach text to grob-name at offset offset (use like \texttt{\texttt{\textbackslash once}})

\texttt{\texttt{\textbackslash balloonText}} [post event] - offset (pair of numbers) text (markup)

Attach text at offset (use like \texttt{\texttt{\textbackslash tweak}})

\texttt{\texttt{\textbackslash bar}} [music] - type (string)

Insert a bar line of type type

\texttt{\texttt{\textbackslash barNumberCheck}} [music] - n (integer)

Print a warning if the current bar number is not n.

\texttt{\texttt{\textbackslash beamExceptions}} (any type) - music (music)

Extract a value suitable for setting Timing.beamExceptions from the given pattern with explicit beams in music. A bar check | has to be used between bars of patterns in order to reset the timing.

\texttt{\texttt{\textbackslash bendAfter}} [post event] - delta (real number)

Create a fall or doit of pitch interval delta.

\texttt{\texttt{\textbackslash bookOutputName}} [void] - newfilename (string)

Direct output for the current book block to newfilename.
\bookOutputSuffix [void] - newsuffix (string)
   Set the output filename suffix for the current book block to newsuffix.

\breathe [music]
   Insert a breath mark.

\chordRepeats [music] - event-types [list] music (music)
   Walk through music putting the notes of the previous chord into repeat chords, as well as an optional list of event-types such as #'(string-number-event).

\clef [music] - type (string)
   Set the current clef to type.

\compoundMeter [music] - args (pair)
   Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as \compoundMeter #'((3 1 8) (2 4)), and a time signature of (3+2)/8 as \compoundMeter #'((3 2 8)) or shorter \compoundMeter #'(3 2 8).

\compressMMRests [music] - music (music)
   Remove the empty bars created by multi-measure rests, leaving just the first bar containing the MM rest itself.

\crossStaff [music] - notes (music)
   Create cross-staff stems.

\cueClef [music] - type (string)
   Set the current cue clef to type.

\cueClefUnset [music]
   Unset the current cue clef.

\cueDuring [music] - what (string) dir (direction) main-music (music)
   Insert contents of quote what corresponding to main-music, in a CueVoice oriented by dir.

\cueDuringWithClef [music] - what (string) dir (direction) clef (string) main-music (music)
   Insert contents of quote what corresponding to main-music, in a CueVoice oriented by dir.

\deadNote [music] - note (music)
   Print note with a cross-shaped note head.

\defineBarLine [void] - bar (string) glyph-list (list)
   Define bar line settings for bar line bar. The list glyph-list must have three entries which define the appearance at the end of line, at the beginning of the next line, and the span bar, respectively.

\displayLilyMusic [music] - port [output port] music (music)
   Display the LilyPond input representation of music to port, defaulting to the console.

\displayMusic [music] - port [output port] music (music)
   Display the internal representation of music to port, defaulting to the console.

\displayScheme (any type) - port [output port] expr (any type)
   Display the internal representation of expr to port, defaulting to the console.

\dropNote [music] - num (integer) music (music)
   Drop a note of any chords in music, in num position from above.
Appendix A: Notation manual tables

\endSpanners [music] - music (music)
Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

\eventChords [music] - music (music)
Compatibility function wrapping EventChord around isolated rhythmic events occurring since version 2.15.28, after expanding repeat chords ‘q’.

\featherDurations [music] - factor (moment) argument (music)
Adjust durations of music in argument by rational factor.

\finger [post event] - finger (integer or markup)
Apply finger as a fingering indication.

\fixed [music] - pitch (pitch) music (music)
Use the octave of pitch as the default octave for music.

\footnote [music] - mark [markup] offset (pair of numbers) footnote (markup) item (symbol list or music)
Make the markup footnote a footnote on item. The footnote is marked with a markup mark moved by offset with respect to the marked music.

If mark is not given or specified as default, it is replaced by an automatically generated sequence number. If item is a symbol list of form ‘Grob’ or ‘Context.Grob’, then grobs of that type will be marked at the current time step in the given context (default Bottom).

If item is music, the music will get a footnote attached to a grob immediately attached to the event, like \tweak does. For attaching a footnote to an indirectly caused grob, write \single\footnote, use item to specify the grob, and follow it with the music to annotate.

Like with \tweak, if you use a footnote on a following post-event, the \footnote command itself needs to be attached to the preceding note or rest as a post-event with -.

\grace [music] - music (music)
Insert music as grace notes.

\grobdescriptions (any type) - descriptions (list)
Create a context modification from descriptions, a list in the format of all-grob-descriptions.

\harmonicByFret [music] - fret (number) music (music)
Convert music into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at fret.

\harmonicByRatio [music] - ratio (number) music (music)
Convert music into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through ratio.

\harmonicNote [music] - note (music)
Print note with a diamond-shaped note head.

\harmonicOn [music]
Set the default note head style to a diamond-shaped style.

\hide [music] - item (symbol list or music)
Set item’s ‘transparent’ property to #t, making it invisible while still retaining its dimensions.
If \textit{item} is a symbol list of form \texttt{GrobName} or \texttt{Context.GrobName}, the result is an override for the grob name specified by it. If \textit{item} is a music expression, the result is the same music expression with an appropriate tweak applied to it.

\texttt{\textbackslash incipit} [music] - incipit-music (music)
Output \textit{incipit-music} before the main staff as an indication of its appearance in the original music.

\texttt{\textbackslash inherit-acceptability} [void] - to (symbol) from (symbol)
When used in an output definition, will modify all context definitions such that context \textit{to} is accepted as a child by all contexts that also accept \textit{from}.

\texttt{\textbackslash inStaffSegno} [music]
Put the segno variant ‘varsegno’ at this position into the staff, compatible with the repeat command.

\texttt{\textbackslash instrumentSwitch} [music] - name (string)
Switch instrument to \textit{name}, which must be predefined with \texttt{\textbackslash addInstrumentDefinition}.

\texttt{\textbackslash inversion} [music] - around (pitch) to (pitch) music (music)
Invert \textit{music} about \textit{around} and transpose from \textit{around} to \textit{to}.

\texttt{\textbackslash invertChords} [music] - num (integer) music (music)
Invert any chords in \textit{music} into their \textit{num}-th position. (Chord inversions may be directed downwards using negative integers.)

\texttt{\textbackslash keepWithTag} [music] - tags (symbol list or symbol) music (music)
Include only elements of \textit{music} that are tagged with one of the tags in \textit{tags}. \textit{tags} may be either a single symbol or a list of symbols.

Each tag may be declared as a member of at most one tag group (defined with \texttt{\textbackslash tagGroup}). If none of a \textit{music} element’s tags share a tag group with one of the specified \textit{tags}, the element is retained.

\texttt{\textbackslash key} [music] - tonic [pitch] pitch-alist [list of number pairs]
Set key to \textit{tonic} and scale \textit{pitch-alist}. If both are null, just generate \texttt{KeyChangeEvent}.

\texttt{\textbackslash killCues} [music] - music (music)
Remove cue notes from \textit{music}.

\texttt{\textbackslash label} [music] - label (symbol)
Create \textit{label} as a referrable label.

\texttt{\textbackslash language} [void] - language (string)
Set note names for language \textit{language}.

\texttt{\textbackslash languageRestore} [void]
Restore a previously-saved pitchnames alist.

\texttt{\textbackslash languageSaveAndChange} [void] - language (string)
Store the previous pitchnames alist, and set a new one.

\texttt{\textbackslash magnifyMusic} [music] - mag (positive number) music (music)
Magnify the notation of \textit{music} without changing the staff-size, using \textit{mag} as a size factor. Stems, beams, slurs, ties, and horizontal spacing are adjusted automatically.

\texttt{\textbackslash magnifyStaff} [music] - mag (positive number)
Change the size of the staff, adjusting notation size and horizontal spacing automatically, using \textit{mag} as a size factor.
\makeClusters [music] - arg (music)
Display chords in arg as clusters.
\makeDefaultStringTuning [void] - symbol (symbol) pitches (list)
This defines a string tuning symbol via a list of pitches. The symbol also gets registered in defaultStringTunings for documentation purposes.
\mark [music] - label [integer or markup]
Make the music for the \mark command.
\markupMap [music] - path (symbol list or symbol) markupfun (markup-function) music (music)
This applies the given markup function markupfun to all markup music properties matching path in music.
For example,
\new Voice { g'2 c'' }
\addlyrics {
  \markupMap LyricEvent.text
    \markup \with-color #red \etc
    { Oh yes! }
}
\modalInversion [music] - around (pitch) to (pitch) scale (music) music (music)
Invert music about around using scale and transpose from around to to.
\modalTranspose [music] - from (pitch) to (pitch) scale (music) music (music)
Transpose music from pitch from to pitch to using scale.
\musicMap [music] - proc (procedure) mus (music)
Apply proc to mus and all of the music it contains.
\noPageBreak [music]
Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.
\noPageTurn [music]
Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.
\octaveCheck [music] - pitch (pitch)
Octave check.
\offset [music] - property (symbol list or symbol) offsets (any type) item (key list or music)
Offset the default value of property of item by offsets. If item is a string, the result is \override for the specified grob type. If item is a music expression, the result is the same music expression with an appropriate tweak applied.
\omit [music] - item (symbol list or music)
Set item’s ‘stencil’ property to #f, effectively omitting it without taking up space. If item is a symbol list of form GrobName or Context.GrobName, the result is an override for the grob name specified by it. If item is a music expression, the result is the same music expression with an appropriate tweak applied to it.
\once [music] - music (music)
Set once to #t on all layout instruction events in music. This will complain about music with an actual duration. As a special exception, if music might be the result of a \tweak command, no warning will be given in order to allow for \once \propertyTweak to work as both one-time override and proper tweak.
\ottava [music] - octave (integer)
    Set the octavation.

\overrideProperty [music] - grob-property-path (list of indexes or symbols) value (any type)
    Set the grob property specified by grob-property-path to value. grob-property-path is a symbol list of the form Context.GrobName.property or GrobName.property, possibly with subproperties given as well.

As opposed to \override which overrides the context-dependent defaults with which a grob is created, this command uses \Output_property_engraver at the grob knowledge stage. This may be necessary for overriding values set after the initial grob creation.

\overrideTimeSignatureSettings [music] - time-signature (fraction, as pair) base-moment (fraction, as pair) beat-structure (list) beam-exceptions (list)
    Override timeSignatureSettings for time signatures of time-signature to have settings of base-moment, beat-structure, and beam-exceptions.

\pageBreak [music]
    Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

\pageTurn [music]
    Force a page turn between two scores or top-level markups.

\palmMute [music] - note (music)
    Print note with a triangle-shaped note head.

\palmMuteOn [music]
    Set the default note head style to a triangle-shaped style.

\parallelMusic [void] - voice-ids (list) music (music)
    Define parallel music sequences, separated by '|' (bar check signs), and assign them to the identifiers provided in voice-ids.

voice-ids: a list of music identifiers (symbols containing only letters)

music: a music sequence, containing BarChecks as limiting expressions.

Example:
\begin{verbatim}
\parallelMusic A,B,C {
    \c c | \d d | \e e |
    \d d | \e e | \f f |
}\\
A = \{ \c c | \d d \}
B = \{ \d d | \e e \}
C = \{ \e e | \f f \}
\end{verbatim}

The last bar checks in a sequence are not copied to the result in order to facilitate ending the last entry at non-bar boundaries.

\parenthesize [music] - arg (music)
    Tag arg to be parenthesized.

\partCombine [music] - chord-range [pair of numbers] part1 (music) part2 (music)
    Take the music in part1 and part2 and return a music expression containing simultaneous voices, where part1 and part2 are combined into one voice where appropriate.

Optional chord-range sets the distance in steps between notes that may be combined into a chord or unison.
Appendix A: Notation manual tables

\partCombineDown [music] - chord-range [pair of numbers] part1 (music) part2 (music)
Take the music in part1 and part2 and typeset so that they share a staff with stems directed downward.

\partCombineForce [music] - type [symbol]
Override the part-combiner.

\partCombineUp [music] - chord-range [pair of numbers] part1 (music) part2 (music)
Take the music in part1 and part2 and typeset so that they share a staff with stems directed upward.

\partial [music] - dur (duration)
Make a partial measure.

\phrasingSlurDashPattern [music] - dash-fraction (number) dash-period (number)
Set up a custom style of dash pattern for dash-fraction ratio of line to space repeated at dash-period interval for phrasing slurs.

\pitchedTrill [music] - main-note (music) secondary-note (music)
Print a trill with main-note as the main note of the trill and print secondary-note as a stemless note head in parentheses.

\pointAndClickOff [void]
Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

\pointAndClickOn [void]
Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

\pointAndClickTypes [void] - types (symbol list or symbol)
Set a type or list of types (such as #'note-event) for which point-and-click info is generated.

\propertyOverride [music] - grob-property-path (list of indexes or symbols) value (any type)
Set the grob property specified by grob-property-path to value. grob-property-path is a symbol list of the form Context.GrobName.property or GrobName.property, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in \override command.

\propertyRevert [music] - grob-property-path (list of indexes or symbols)
Revert the grob property specified by grob-property-path to its previous value. grob-property-path is a symbol list of the form Context.GrobName.property or GrobName.property, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in \revert command.

\propertySet [music] - property-path (symbol list or symbol) value (any type)
Set the context property specified by property-path to value. This music function is mostly intended for use from Scheme as a substitute for the built-in \set command.

\propertyTweak [music] - prop (key list or symbol) value (any type) item (key list or music)
Add a tweak to the following item, usually music. This generally behaves like \tweak but will turn into an \override when item is a symbol list.
In that case, item specifies the grob path to override. This is mainly useful when using \propertyTweak as a component for building other functions like \omit.
It is not the default behavior for \tweak since many input strings in \lyricmode
can serve equally as music or as symbols which causes surprising behavior when
tweaking lyrics using the less specific semantics of $\text{\textbackslash propertyTweak}$.

$prop$ can contain additional elements in which case a nested property (inside of an
alist) is tweaked.

$\text{\textbackslash propertyUnset}$ [music] - $prop$ (symbol list or symbol)
Unset the context property specified by $prop$. This music function is mostly
intended for use from Scheme as a substitute for the built-in $\text{\textbackslash unset}$ command.

$\text{\textbackslash pushToTag}$ [music] - $tag$ (symbol) $more$ (music) $music$ (music)
Add $more$ to the front of $\text{\textbackslash elements}$ of all music expressions in $music$ that are tagged
with $tag$.

$\text{\textbackslash quoteDuring}$ [music] - $what$ (string) $main-music$ (music)
Indicate a section of music to be quoted. $what$ indicates the name of the quoted
voice, as specified in an $\text{\textbackslash addQuote}$ command. $main-music$ is used to indicate the
length of music to be quoted; usually contains spacers or multi-measure rests.

$\text{\textbackslash raiseNote}$ [music] - $num$ (integer) $music$ (music)
Raise a note of any chords in $music$, in $num$ position from below.

$\text{\textbackslash reduceChords}$ [music] - $music$ (music)
Reduce chords contained in $music$ to single notes, intended mainly for reusing music
in RhythmicStaff. Does not reduce parallel music.

$\text{\textbackslash relative}$ [music] - $pitch$ [pitch] $music$ (music)
Make $music$ relative to $pitch$. If $pitch$ is omitted, the first note in $music$ is given in
absolute pitch.

$\text{\textbackslash removeWithTag}$ [music] - $tags$ (symbol list or symbol) $music$ (music)
Remove elements of $music$ that are tagged with one of the tags in $tags$. $tags$ may
be either a single symbol or a list of symbols.

$\text{\textbackslash resetRelativeOctave}$ [music] - $pitch$ (pitch)
Set the octave inside a $\text{\textbackslash relative}$ section.

$\text{\textbackslash retrograde}$ [music] - $music$ (music)
Return $music$ in reverse order.

$\text{\textbackslash revertTimeSignatureSettings}$ [music] - $time-signature$ (pair)
Revert $timeSignatureSettings$ for time signatures of $time-signature$.

$\text{\textbackslash rightHandFinger}$ [post event] - $finger$ (integer or markup)
Apply $finger$ as a fingering indication.

$\text{\textbackslash scaleDurations}$ [music] - $fraction$ (non-negative rational, fraction, or moment) $music$ (music)
Multiply the duration of events in $music$ by $fraction$.

$\text{\textbackslash settingsFrom}$ (any type) - $ctx$ [symbol] $music$ (music)
Take the layout instruction events from $music$, optionally restricted to those applying
to context type $ctx$, and return a context modification duplicating their effect.

$\text{\textbackslash shape}$ [music] - $offsets$ (list) $item$ (key list or music)
Offset control-points of $item$ by $offsets$. The argument is a list of number pairs or
list of such lists. Each element of a pair represents an offset to one of the coordinates
of a control-point. If $item$ is a string, the result is $\text{\textbackslash once\textbackslash override}$ for the specified
grob type. If $item$ is a music expression, the result is the same music expression
with an appropriate tweak applied.
\shiftDurations [music] - \( \text{dur} \) (integer) \( \text{dots} \) (integer) \( \text{arg} \) (music)
Change the duration of \( \text{arg} \) by adding \( \text{dur} \) to the \text{durlog} of \( \text{arg} \) and \( \text{dots} \) to the \text{dots} of \( \text{arg} \).

\single [music] - \text{overrides} (music) \text{music} (music)
Convert \text{overrides} to tweaks and apply them to \text{music}. This does not convert \text{\revert}, \text{\set} or \text{\unset}.

\skip [music] - \( \text{dur} \) (duration)
Skip forward by \( \text{dur} \).

\slashedGrace [music] - \text{music} (music)
Create slashed graces (slashes through stems, but no slur) from the following music expression

\slurDashPattern [music] - \text{dash-fraction} (number) \text{dash-period} (number)
Set up a custom style of dash pattern for \text{dash-fraction} ratio of line to space repeated at \text{dash-period} interval for slurs.

\spacingTweaks [music] - \text{parameters} (list)
Set the system stretch, by reading the ‘system-stretch property of the ‘parameters’ assoc list.

\storePredefinedDiagram [void] - \text{fretboard-table} (hash table) \text{chord} (music) \text{tuning} (pair) \text{diagram-definition} (string or pair)
Add predefined fret diagram defined by \text{diagram-definition} for the chord pitches \text{chord} and the string\text{Tuning} \text{tuning}.

\stringTuning (any type) - \text{chord} (music)
Convert \text{chord} to a string tuning. \text{chord} must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

\styledNoteHeads [music] - \text{style} (symbol) \text{heads} (symbol list or symbol) \text{music} (music)
Set \text{heads} in \text{music} to \text{style}.

\tabChordRepeats [music] - \text{event-types} (list) \text{music} (music)
Walk through \text{music} putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of \text{event-types} such as \#'\text{(articulation-event)}.

\tabChordRepetition [void]
Include the string and fingering information in a chord repetition. This function is deprecated; try using \text{\tabChordRepeats} instead.

\tag [music] - \text{tags} (symbol list or symbol) \text{music} (music)
Tag the following \text{music} with \text{tags} and return the result, by adding the single symbol or symbol list \text{tags} to the \text{tags} property of \text{music}.

\tagGroup [void] - \text{tags} (symbol list)
Define a tag group comprising the symbols in the symbol list \text{tags}. Tag groups must not overlap.

\temporary [music] - \text{music} (music)
Make any \text{\override} in \text{music} replace an existing grob property value only temporarily, restoring the old value when a corresponding \text{\revert} is executed. This is achieved by clearing the ‘pop-first’ property normally set on \text{\overrides}.
An \text{\override}/\text{\revert} sequence created by using \text{\temporary} and \text{\undo} on the same music containing overrides will cancel out perfectly or cause a warning.
Non-property-related music is ignored, warnings are generated for any property-changing music that isn’t an \text{\override}.
Appendix A: Notation manual tables

\tieDashPattern [music] - dash-fraction (number) dash-period (number)
Set up a custom style of dash pattern for dash-fraction ratio of line to space repeated at dash-period interval for ties.

\time [music] - beat-structure [number list] fraction (fraction, as pair)
Set fraction as time signature, with optional number list beat-structure before it.

\times [music] - fraction (fraction, as pair) music (music)
Scale music in time by fraction.

\tocItem [music] - label [symbol list or symbol] text (markup)
Add a line to the table of contents, using the tocItemMarkup paper variable markup and assigning it to label if one is provided. If a hierarchy of labels is given, make the current item a child of the corresponding objects.

\transpose [music] - from (pitch) to (pitch) music (music)
Transpose music from pitch from to pitch to.

\transposedCueDuring [music] - what (string) dir (direction) pitch (pitch) main-music (music)
Insert notes from the part what into a voice called cue, using the transposition defined by pitch. This happens simultaneously with main-music, which is usually a rest. The argument dir determines whether the cue notes should be notated as a first or second voice.

\transposition [music] - pitch (pitch)
Set instrument transposition

\tuplet [music] - ratio (fraction, as pair) tuplet-span [duration] music (music)
Scale the given music to tuplets. ratio is a fraction that specifies how many notes are played in place of the nominal value: it will be ‘3/2’ for triplets, namely three notes being played in place of two. If the optional duration tuplet-span is specified, it is used instead of tupletSpannerDuration for grouping the tuplets. For example,

\tuplet 3/2 4 { c8 c c c c c }
will result in two groups of three tuplets, each group lasting for a quarter note.

\tupletSpan [music] - tuplet-span [duration]
Set tupletSpannerDuration, the length into which \tuplet without an explicit ‘tuplet-span' argument of its own will group its tuplets, to the duration tuplet-span. To revert to the default of not subdividing the contents of a \tuplet command without explicit ‘tuplet-span', use

\tupletSpan \default

\tweak [music] - prop (key list or symbol) value (any type) music (music)
Add a tweak to the following music. Layout objects created by music get their property prop set to value. If prop has the form ‘Grob.property', like with

\tweak Accidental.color #red cis'
an indirectly created grob (‘Accidental' is caused by ‘NoteHead') can be tweaked; otherwise only directly created grobs are affected.
prop can contain additional elements in which case a nested property (inside of an alist) is tweaked.
If music is an ‘event-chord', every contained rhythmic-event is tweaked instead.

\undo [music] - music (music)
Convert \override and \set in music to \revert and \unset, respectively. Any reverts and unsets already in music cause a warning. Non-property-related music is ignored.
\unfoldRepeats [music] - types [symbol list or symbol] music (music)
Force \repeat volta, \repeat tremolo or \repeat percent commands in music to be interpreted as \repeat unfold, if specified in the optional symbol-list \textit{types}. The default for \textit{types} is an empty list, which will force any of those commands in \textit{music} to be interpreted as \repeat unfold. Possible entries are \textit{volta}, \textit{tremolo} or \textit{percent}. Multiple entries are possible.

\voices [music] - ids (list of indexes or symbols) music (music)
Take the given key list of numbers (indicating the use of \\textit{\voiceOne}...) or symbols (indicating voice names, typically converted from strings by argument list processing) and assign the following \textbackslash-separated music to contexts according to that list. Named rather than numbered contexts can be used for continuing one voice (for the sake of spanners and lyrics), usually requiring a \textit{\voiceOne}-style override at the beginning of the passage and a \texttt{oneVoice} override at its end.

The default
\begin{verbatim}
<< ... \ \ \ ... \ \ ... >>
\end{verbatim}

construct would correspond to
\begin{verbatim}
\voices 1,2,3 << ... \ \ ... \ \ ... >>
\end{verbatim}

\void [void] - arg (any type)
Accept a scheme argument, return a void expression. Use this if you want to have a scheme expression evaluated because of its side-effects, but its value ignored.

\withMusicProperty [music] - sym (symbol) val (any type) music (music)
Set \textit{sym} to \textit{val} in \textit{music}.

\xNote [music] - note (music)
Print \textit{note} with a cross-shaped note head.

\= [post event] - id (index or symbol) event (post event)
This sets the \textit{spanner-id} property of the following \textit{event} to the given \textit{id} (non-negative integer or symbol). This can be used to tell LilyPond how to connect overlapping or parallel slurs or phrasing slurs within a single \texttt{Voice}.

\begin{verbatim}
\fixed c' { c\=1( d\=2( e\=1) f\=2) }
\end{verbatim}

\begin{center}
\includegraphics[width=0.2\textwidth]{example.png}
\end{center}

\section{A.20 Context modification identifiers}
The following commands are defined for use as context modifications within a \texttt{\layout} or \texttt{\with} block.

\RemoveAllEmptyStaves
Remove staves which are considered to be empty according to the list of interfaces set by \texttt{keepAliveInterfaces}, including those in the first system.

\begin{itemize}
\item Sets grob property \texttt{remove-empty} in Section ‘‘VerticalAxisGroup’’ in Internals Reference to \#t.
\item Sets grob property \texttt{remove-first} in Section ‘‘VerticalAxisGroup’’ in Internals Reference to \#t.
\end{itemize}
\RemoveEmptyStaves
Remove staves which are considered to be empty according to the list of interfaces set by \texttt{keepAliveInterfaces}.

- Sets grob property \texttt{remove-empty} in Section ‘‘VerticalAxisGroup’’ in Internals Reference to \texttt{#t}.

\section{Predefined type predicates}\label{appendix:a:predefined-type-predicates}
Predicates return \texttt{#t} when their argument is of the named type and \texttt{#f} if it isn’t.

\subsection*{R5RS primary predicates}
Primary predicates can be applied to any expression. They can be used on their own as predicates for LilyPond functions. The predicates here are part of the Scheme standard R5RS.

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean?</td>
<td>boolean</td>
</tr>
<tr>
<td>char?</td>
<td>character</td>
</tr>
<tr>
<td>complex?</td>
<td>complex number</td>
</tr>
<tr>
<td>eof-object?</td>
<td>end-of-file object</td>
</tr>
<tr>
<td>input-port?</td>
<td>input port</td>
</tr>
<tr>
<td>integer?</td>
<td>integer</td>
</tr>
<tr>
<td>list?</td>
<td>list (use \texttt{cheap-list?} for faster processing)</td>
</tr>
<tr>
<td>null?</td>
<td>null</td>
</tr>
<tr>
<td>number?</td>
<td>number</td>
</tr>
<tr>
<td>output-port?</td>
<td>output port</td>
</tr>
<tr>
<td>pair?</td>
<td>pair</td>
</tr>
<tr>
<td>port?</td>
<td>port</td>
</tr>
<tr>
<td>procedure?</td>
<td>procedure</td>
</tr>
<tr>
<td>rational?</td>
<td>rational number</td>
</tr>
<tr>
<td>real?</td>
<td>real number</td>
</tr>
<tr>
<td>string?</td>
<td>string</td>
</tr>
<tr>
<td>symbol?</td>
<td>symbol</td>
</tr>
<tr>
<td>vector?</td>
<td>vector</td>
</tr>
</tbody>
</table>

\subsection*{R5RS secondary predicates}
Secondary predicates are only applicable to specific expressions (for example, to numbers). They will throw a type error when applied to expressions they are not intended for. The predicates here are part of the Scheme standard R5RS.

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char-alphabetic?</td>
<td>alphabetic character</td>
</tr>
<tr>
<td>char-lower-case?</td>
<td>lower-case character</td>
</tr>
<tr>
<td>char-numeric?</td>
<td>numeric character</td>
</tr>
<tr>
<td>char-upper-case?</td>
<td>upper-case character</td>
</tr>
<tr>
<td>char-whitespace?</td>
<td>whitespace character</td>
</tr>
<tr>
<td>even?</td>
<td>even number</td>
</tr>
<tr>
<td>exact?</td>
<td>exact number</td>
</tr>
<tr>
<td>inexact?</td>
<td>inexact number</td>
</tr>
<tr>
<td>negative?</td>
<td>negative number</td>
</tr>
<tr>
<td>odd?</td>
<td>odd number</td>
</tr>
<tr>
<td>positive?</td>
<td>positive number</td>
</tr>
<tr>
<td>zero?</td>
<td>zero</td>
</tr>
</tbody>
</table>
**Guile predicates**

These predicates are defined by Guile but are not part of a Scheme standard.

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash-table?</td>
<td>hash table</td>
</tr>
</tbody>
</table>

**LilyPond scheme predicates**

These predicates are only available within LilyPond and defined in Scheme.

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean-or-symbol?</td>
<td>boolean or symbol</td>
</tr>
<tr>
<td>cheap-list?</td>
<td>list <em>(use this instead of list? for faster processing)</em></td>
</tr>
<tr>
<td>color?</td>
<td>color</td>
</tr>
<tr>
<td>fraction?</td>
<td>fraction, as pair</td>
</tr>
<tr>
<td>grob-list?</td>
<td>list of grobs</td>
</tr>
<tr>
<td>index?</td>
<td>non-negative integer</td>
</tr>
<tr>
<td>integer-or-markup?</td>
<td>integer or markup</td>
</tr>
<tr>
<td>key?</td>
<td>index or symbol</td>
</tr>
<tr>
<td>key-list?</td>
<td>list of indexes or symbols</td>
</tr>
<tr>
<td>key-list-or-music?</td>
<td>key list or music</td>
</tr>
<tr>
<td>key-list-or-symbol?</td>
<td>key list or symbol</td>
</tr>
<tr>
<td>markup?</td>
<td>markup</td>
</tr>
<tr>
<td>markup-command-list?</td>
<td>markup command list</td>
</tr>
<tr>
<td>markup-list?</td>
<td>markup list</td>
</tr>
<tr>
<td>moment-pair?</td>
<td>pair of moment objects</td>
</tr>
<tr>
<td>number-list?</td>
<td>number list</td>
</tr>
<tr>
<td>number-or-grob?</td>
<td>number or grob</td>
</tr>
<tr>
<td>number-or-pair?</td>
<td>number or pair</td>
</tr>
<tr>
<td>number-or-string?</td>
<td>number or string</td>
</tr>
<tr>
<td>number-pair?</td>
<td>pair of numbers</td>
</tr>
<tr>
<td>number-pair-list?</td>
<td>list of number pairs</td>
</tr>
<tr>
<td>rational-or-procedure?</td>
<td>an exact rational or procedure</td>
</tr>
<tr>
<td>rhythmic-location?</td>
<td>rhythmic location</td>
</tr>
<tr>
<td>scale?</td>
<td>non-negative rational, fraction, or moment</td>
</tr>
<tr>
<td>scheme?</td>
<td>any type</td>
</tr>
<tr>
<td>string-or-music?</td>
<td>string or music</td>
</tr>
<tr>
<td>string-or-pair?</td>
<td>string or pair</td>
</tr>
<tr>
<td>string-or-symbol?</td>
<td>string or symbol</td>
</tr>
<tr>
<td>symbol-list?</td>
<td>symbol list</td>
</tr>
<tr>
<td>symbol-list-or-music?</td>
<td>symbol list or music</td>
</tr>
<tr>
<td>symbol-list-or-symbol?</td>
<td>symbol list or symbol</td>
</tr>
<tr>
<td>void?</td>
<td>void</td>
</tr>
</tbody>
</table>

**LilyPond exported predicates**

These predicates are only available within LilyPond and usually defined in C++.

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ly:box?</td>
<td>box</td>
</tr>
<tr>
<td>ly:context?</td>
<td>context</td>
</tr>
<tr>
<td>ly:context-def?</td>
<td>context definition</td>
</tr>
</tbody>
</table>
A.22 Scheme functions

ly:add-context-mod contextmods modification

[Function]
Adds the given context modification to the list contextmods of context modifications.

ly:add-file-name-alist alist

[Function]
Add mappings for error messages from alist.
**ly:add-interface** *iface desc props*  
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

**ly:add-listener** *callback disp cl*  
Add the single-argument procedure *callback* as listener to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it calls *callback* with it.

**ly:add-option** *sym val description*  
Add a program option *sym*. *val* is the default value and *description* is a string description.

**ly:all-grob-interfaces**  
Return the hash table with all grob interface descriptions.

**ly:all-options**  
Get all option settings in an alist.

**ly:all-stencil-expressions**  
Return all symbols recognized as stencil expressions.

**ly:angle** *x y*  
Calculates angle in degrees of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

**ly:assoc-get** *key alist default-value strict-checking*  
Return value if *key* in *alist*, else *default-value* (or #f if not specified). If *strict-checking* is set to #t and *key* is not in *alist*, a programming_error is output.

**ly:axis-group-interface::add-element** *grob grob-element*  
Set *grob* the parent of *grob-element* on all axes of *grob*.

**ly:basic-progress** *str rest*  
A Scheme callable function to issue a basic progress message *str*. The message is formatted with format and *rest*.

**ly:book-score-count**  
count number of beam scores.

**ly:book?** *x*  
Is *x* a Book object?


**ly:book-add-score!** *book-smob score*  
Add score to *book-smob* score list.

**ly:book-book-parts** *book*  
Return book parts in *book*.

**ly:book-header** *book*  
Return header in *book*.

**ly:book-paper** *book*  
Return paper in *book*.

**ly:book-process** *book-smob default-paper default-layout output*  
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).
Print book. output is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

ly:book-scores book
Return scores in book.

Set the book header.

ly:box? x
Is x a Box object?

ly:bp num
num bigpoints (1/72th inch).

ly:bracket a iv t p
Make a bracket in direction a. The extent of the bracket is given by iv. The wings protrude by an amount of p, which may be negative. The thickness is given by t.

ly:broadcast disp ev
Send the stream event ev to the dispatcher disp.

ly:camel-case->lisp-identifier name-sym
Convert FooBar_Bla to foo-bar-bla style symbol.

ly:chain-assoc-get key achain default-value 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.

ly:check-expected-warnings
Check whether all expected warnings have really been triggered.

ly:cm num
num cm.

ly:command-line-code
The Scheme code specified on command-line with -e.

ly:command-line-options
The Scheme options specified on command-line with -d.

ly:connect-dispatchers to from
Make the dispatcher to listen to events from from.

ly:context? x
Is x a Context object?

ly:context-current-moment context
Return the current moment of context.

ly:context-def? x
Is x a Context_def object?
Appendix A: Notation manual tables

ly:context-def-lookup def sym val
   [Function]
   Return the value of sym in context definition def (e.g., Voice). If no value is found, return val or '()' if val is undefined. sym can be any of 'default-child', 'consists', 'description', 'aliases', 'accepts', 'property-ops', 'context-name', 'group-type'.

ly:context-def-modify def mod
   [Function]
   Return the result of applying the context-mod mod to the context definition def. Does not change def.

ly:context-event-source context
   [Function]
   Return event-source of context context.

ly:context-events-below context
   [Function]
   Return a stream-distributor that distributes all events from context and all its subcontexts.

ly:context-find context name
   [Function]
   Find a parent of context that has name or alias name. Return #f if not found.

ly:context-grob-definition context name
   [Function]
   Return the definition of name (a symbol) within context as an alist.

ly:context-id context
   [Function]
   Return the ID string of context, i.e., for \context Voice = "one" ... return the string one.

ly:context-matched-pop-property context grob cell
   [Function]
   This undoes a particular \override, \once \override or \once \revert when given the specific alist pair to undo.

ly:context-mod? x
   [Function]
   Is x a Context_mod object?

ly:context-mod-apply! context mod
   [Function]
   Apply the context modification mod to context.

ly:context-name context
   [Function]
   Return the name of context, i.e., for \context Voice = "one" ... return the symbol Voice.

ly:context-now context
   [Function]
   Return now-moment of context context.

ly:context-parent context
   [Function]
   Return the parent of context, #f if none.

ly:context-property context sym def
   [Function]
   Return the value for property sym in context. If def is given, and property value is '()', return def.

ly:context-property-where-defined context name
   [Function]
   Return the context above context where name is defined.

ly:context-pushpop-property context grob eltprop val
   [Function]
   Do \temporary \override or \revert operation in context. The grob definition grob is extended with eltprop (if val is specified) or reverted (if unspecified).

ly:context-set-property! context name val
   [Function]
   Set value of property name in context context to val.
**ly:context-unset-property** *context name*
Unset value of property *name* in context *context*.

**ly:debug** *str rest*
A Scheme callable function to issue a debug message *str*. The message is formatted with *format* and *rest*.

**ly:default-scale**
Get the global default scale.

**ly:dimension?** *d*
Is *d* a dimension? Used to distinguish length variables from normal numbers.

**ly:dir?** *s*
Is *s* a direction? Valid directions are -1, 0, or 1, where -1 represents left or down, 1 represents right or up, and 0 represents a neutral direction.

**ly:directed** *direction magnitude*
Calculates an \((x \cdot y)\) pair with optional *magnitude* (defaulting to 1.0) and *direction* specified either as an angle in degrees or a coordinate pair giving the direction. If *magnitude* is a pair, the respective coordinates are scaled independently, useful for ellipse drawings.

**ly:disconnect-dispatchers** *to from*
Stop the dispatcher *to* listening to events from *from*.

**ly:dispatcher?** *x*
Is *x* a Dispatcher object?

**ly:duration?** *x*
Is *x* a Duration object?

**ly:duration<?** *p1 p2*
Is *p1* shorter than *p2*?

**ly:duration->string** *dur*
Convert *dur* to a string.

**ly:duration-dot-count** *dur*
Extract the dot count from *dur*.

**ly:duration-factor** *dur*
Extract the compression factor from *dur*. Return it as a pair.

**ly:duration-length** *dur*
The length of the duration as a moment.

**ly:duration-log** *dur*
Extract the duration log from *dur*.

**ly:duration-scale** *dur*
Extract the compression factor from *dur*. Return it as a rational.

**ly:effective-prefix**
Return effective prefix.

**ly:encode-string-for-pdf** *str*
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that’s not possible to full UTF-16BE with Byte-Order-Mark (BOM).
ly:engraver-announce-end-grob engraver grob cause  
Announce the end of a grob (i.e., the end of a spanner) originating from given engraver 
instance, with grob being a grob. cause should either be another grob or a music event.

ly:engraver-make-grob engraver grob-name cause  
Create a grob originating from given engraver instance, with given grob-name, a symbol. 
cause should either be another grob or a music event.

ly:error str rest  
A Scheme callable function to issue the error str. The error is formatted with format and 
rest.

ly:event? obj  
Is obj a proper (non-rhythmic) event object?

ly:event-deep-copy m  
Copy m and all sub expressions of m.

ly:event-property sev sym val  
Get the property sym of stream event sev. If sym is undefined, return val or '()' if val is not 
specified.

ly:event-set-property! ev sym val  
Set property sym in event ev to val.

ly:expand-environment str  
Expand $VAR and ${VAR} in str.

ly:expect-warning str rest  
A Scheme callable function to register a warning to be expected and subsequently suppressed. 
If the warning is not encountered, a warning about the missing warning will be shown. The 
message should be translated with (_ ...) and changing parameters given after the format 
string.

ly:extract-subfont-from-collection collection-file-name idx subfont-file-name  
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.

ly:find-file name  
Return the absolute file name of name, or #f if not found.

ly:font-config-add-directory dir  
Add directory dir to FontConfig.

ly:font-config-add-font font  
Add font font to FontConfig.

ly:font-config-display-fonts  
Dump a list of all fonts visible to FontConfig.

ly:font-config-get-font-file name  
Get the file for font name.

ly:font-design-size font  
Given the font metric font, return the design size, relative to the current output-scale.
**ly:font-file-name** *font*  
Given the font metric *font*, return the corresponding file name.

**ly:font-get-glyph** *font* *name*  
Given the font metric *font*, return the stencil for the glyph named *name*. If the glyph is not available, return an empty stencil.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

**ly:font-glyph-name-to-charcode** *font* *name*  
Return the character code for glyph *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

**ly:font-glyph-name-to-index** *font* *name*  
Return the index for *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

**ly:font-index-to-charcode** *font* *index*  
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*  
Given the font metric *font*, return the magnification, relative to the current output-scale.

**ly:font-metric?** *x*  
Is *x* a `Font_metric` object?

**ly:font-name** *font*  
Given the font metric *font*, return the corresponding name.

**ly:font-sub-fonts** *font*  
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

**ly:format** *str* *rest*  
LilyPond specific format, supporting \~a and \~[0-9]f. Basic support for \~s is also provided.

**ly:format-output** *context*  
Given a global context in its final state, process it and return the `Music_output` object in its final state.

**ly:generic-bound-extent** *grob* *common*  
Determine the extent of *grob* relative to *common* along the X axis, finding its extent as a bound when it a has `bound-alignment-interfaces` property list set and otherwise the full extent.

**ly:get-all-function-documentation**  
Get a hash table with all LilyPond Scheme extension functions.
ly:get-all-translators  [Function]
Return a list of all translator objects that may be instantiated.

ly:get-cff-offset font-file-name idx  [Function]
Get the offset of 'CFF' table for font_file_name, returning it as an integer. The optional idx argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of idx is 0.

ly:get-context-mods contextmod  [Function]
Returns the list of context modifications stored in contextmod.

ly:get-font-format font-file-name idx  [Function]
Get the font format for font_file_name, returning it as a symbol. The optional idx argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of idx is 0.

ly:get-option var  [Function]
Get a global option setting.

ly:get-spacing-spec from-scm to-scm  [Function]
Return the spacing spec going between the two given grobs, from_scm and to_scm.

ly:get-undead undead  [Function]
Get back object from undead.

ly:gettext original  [Function]
A Scheme wrapper function for gettext.

ly:grob? x  [Function]
Is x a Grob object?

ly:grob-alist-chain grob global  [Function]
Get an alist chain for grob grob, with global as the global default. If unspecified, font-defaults from the layout block is taken.

ly:grob-array? x  [Function]
Is x a Grob_array object?

ly:grob-array->list grob-arr  [Function]
Return the elements of grob-arr as a Scheme list.

ly:grob-array-length grob-arr  [Function]
Return the length of grob-arr.

ly:grob-array-ref grob-arr index  [Function]
Retrieve the indexth element of grob-arr.

ly:grob-basic-properties grob  [Function]
Get the immutable properties of grob.

ly:grob-chain-callback grob proc sym  [Function]
Find the callback that is stored as property sym of grob grob and chain proc to the head of this, meaning that it is called using grob and the previous callback's result.

ly:grob-common-refpoint grob other axis  [Function]
Find the common refpoint of grob and other axis.
ly:grob-common-refpoint-of-array grob others axis
    Find the common refpoint of grob and others (a grob-array) for axis.

ly:grob-default-font grob
    Return the default font for grob grob.

ly:grob-extent grob refp axis
    Get the extent in axis direction of grob relative to the grob refp.

ly:grob-get-vertical-axis-group-index grob
    Get the index of the vertical axis group the grob grob belongs to; return -1 if none is found.

ly:grob-interfaces grob
    Return the interfaces list of grob grob.

ly:grob-layout grob
    Get \layout definition from grob grob.

ly:grob-object grob sym
    Return the value of a pointer in grob grob of property sym. It returns '()' (end-of-list) if sym is undefined in grob.

ly:grob-original grob
    Return the unbroken original grob of grob.

ly:grob-parent grob axis
    Get the parent of grob. axis is 0 for the X-axis, 1 for the Y-axis.

ly:grob-pq<?, a, b
    Compare two grob priority queue entries. This is an internal function.

ly:grob-properties? x
    Is x a Grob_properties object?

ly:grob-property grob sym val
    Return the value for property sym of grob. If no value is found, return val or '()' if val is not specified.

ly:grob-property-data grob sym
    Return the value for property sym of grob, but do not process callbacks.

ly:grob-pure-height grob refp beg end val
    Return the pure height of grob given refpoint refp. If no value is found, return val or '()' if val is not specified.

ly:grob-pure-property grob sym beg end val
    Return the pure value for property sym of grob. If no value is found, return val or '()' if val is not specified.

ly:grob-relative-coordinate grob refp axis
    Get the coordinate in axis direction of grob relative to the grob refp.

ly:grob-robust-relative-extent grob refp axis
    Get the extent in axis direction of grob relative to the grob refp, or (0,0) if empty.

ly:grob-script-priority-less a, b
    Compare two grobs by script priority. For internal use.
ly:grob-set-nested-property! \( \text{grob symlist val} \)  
Set nested property symlist in grob grob to value val.

ly:grob-set-object! \( \text{grob sym val} \)  
Set sym in grob grob to value val.

ly:grob-set-parent! \( \text{grob axis parent-grob} \)
Set parent-grob the parent of grob grob in axis axis.

ly:grob-set-property! \( \text{grob sym val} \)  
Set sym in grob grob to value val.

ly:grob-spanned-rank-interval \( \text{grob} \)
Returns a pair with the rank of the furthest left column and the rank of the furthest right column spanned by grob.

ly:grob-staff-position sg
Return the Y-position of sg relative to the staff.

ly:grob-suicide! \( \text{grob} \)
Kill grob.

ly:grob-system \( \text{grob} \)
Return the system grob of grob.

ly:grob-translate-axis! \( \text{grob d a} \)
Translate grob on axis a over distance d.

ly:grob-vertical<? \( \text{a b} \)
Does a lie above b on the page?

ly:gulp-file \( \text{name size} \)
Read size characters from the file name, and return its contents in a string. If size is undefined, the entire file is read. The file is looked up using the search path.

ly:has-glyph-names? \( \text{font-file-name idx} \)
Does the font for font file name have glyph names? The optional idx argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of idx is 0.

ly:hash-table-keys \( \text{tab} \)
Return a list of keys in tab.

ly:inch \( \text{num} \)
num inches.

ly:input-both-locations sip
Return input location in sip as (file-name first-line first-column last-line last-column).

ly:input-file-line-char-column sip
Return input location in sip as (file-name line char column).

ly:input-location? \( \text{x} \)
Is x a Input object?

ly:input-message sip msg rest
Print msg as a GNU compliant error message, pointing to the location in sip. msg is interpreted similar to format's argument, using rest.
**ly:input-warning** \( \text{sip msg rest} \)  
Print \( \text{msg} \) as a GNU compliant warning message, pointing to the location in \( \text{sip} \). \( \text{msg} \) is interpreted similar to \text{format}'s argument, using \( \text{rest} \).

**ly:interpret-music-expression** \( \text{mus ctx} \)  
Interpret the music expression \( \text{mus} \) in the global context \( \text{ctx} \). The context is returned in its final state.

**ly:intlog2** \( d \)  
The 2-logarithm of \( 1/d \).

**ly:item?** \( g \)  
Is \( g \) an \text{Item} object?

**ly:item-break-dir** \( it \)  
The break status direction of item \( it \). -1 means end of line, 0 unbroken, and 1 beginning of line.

**ly:item-get-column** \( it \)  
Return the \text{PaperColumn} or \text{NonMusicalPaperColumn} associated with this \text{Item}.

**ly:iterator?** \( x \)  
Is \( x \) a \text{Music_iterator} object?

**ly:length** \( x y \)  
Calculates magnitude of given vector. With one argument, \( x \) is a number pair indicating the vector. With two arguments, \( x \) and \( y \) specify the respective coordinates.

**ly:lily-lexer?** \( x \)  
Is \( x \) a \text{Lily_lexer} object?

**ly:lily-parser?** \( x \)  
Is \( x \) a \text{Lily_parser} object?

**ly:line-interface::line** \( \text{grob startx starty endx endy} \)  
Make a line using layout information from \( \text{grob} \).

**ly:listened-event-class?** \( \text{disp cl} \)  
Does \( \text{disp} \) listen to any event type in the list \( \text{cl} \)?

**ly:listened-event-types** \( \text{disp} \)  
Return a list of all event types that \( \text{disp} \) listens to.

**ly:listener?** \( x \)  
Is \( x \) a \text{Listener} object?

**ly:make-book** \( \text{paper header scores} \)  
Make a \textbackslash book of \( \text{paper} \) and \( \text{header} \) (which may be \#f as well) containing \textbackslash scores.

**ly:make-book-part** \( \text{scores} \)  
Make a \textbackslash bookpart containing \textbackslash scores.

**ly:make-context-mod** \( \text{mod-list} \)  
Creates a context modification, optionally initialized via the list of modifications \( \text{mod-list} \).

**ly:make-dispatcher**  
Return a newly created dispatcher.
ly:make-duration length dotcount num den

length is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument dotcount.

The duration factor is optionally given by integers num and den, alternatively by a single rational number.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

ly:make-global-context output-def

Set up a global interpretation context, using the output block output-def. The context is returned.

ly:make-global-translator global

Create a translator group and connect it to the global context global. The translator group is returned.

ly:make-grob-properties alist

This packages the given property list alist in a grob property container stored in a context property with the name of a grob.

ly:make-moment m g gn gd

Create the moment with rational main timing m, and optional grace timing g.

A moment is a point in musical time. It consists of a pair of rationals (m, g), where m is the timing for the main notes, and g the timing for grace notes. In absence of grace notes, g is zero.

For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.

ly:make-music props

Make a C++ Music object and initialize it with props.

This function is for internal use and is only called by make-music, which is the preferred interface for creating music objects.

ly:make-music-function signature func

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

Make music relative to pitch, return final pitch.

ly:make-output-def

Make an output definition.

ly:make-page-label-marker label

Return page marker with label label.

ly:make-page-permission-marker symbol permission

Return page marker with page breaking and turning permissions.

ly:make-pango-description-string chain size

Make a PangoFontDescription string for the property alist chain at size size.
### Appendix A: Notation manual tables

**ly:make-paper-outputter** *port* *alist* *default-callback*  
Create an outputter dumping to *port*. *alist* should map symbols to procedures. See [output-ps.scm](#) for an example. If *default-callback* is given, it is called for unsupported expressions.

**ly:make-pitch** *octave* *note* *alter*  
*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*  
Create a Prob object.

**ly:make-rotation** *angle* *center*  
Make a transform rotating by *angle* in degrees. If *center* is given as a pair of coordinates, it is the center of the rotation, otherwise the rotation is around (0.0).

**ly:make-scale** *steps*  
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

**ly:make-scaling** *scale* *scaley*  
Create a scaling transform from argument *scale* and optionally *scaley*. When both arguments are given, they must be real and give the scale in x and y direction. If only *scale* is given, it may also be complex to indicate a scaled rotation in the manner of complex number rotations, or a pair of reals for specifying different scales in x and y direction like with the first calling convention.

**ly:make-score** *music*  
Return score with *music* encapsulated in it.

**ly:make-spring** *ideal* *min-dist*  
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.

**ly:make-stencil** *expr* *xext* *yext*  
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example [scm/output-ps.scm](#).
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use *empty-interval* as its value), it is taken to be empty.

**ly:make-stream-event** *cl* *proplist*  
Create a stream event of class *cl* with the given mutable property list.

**ly:make-transform** *xx* *yx* *xy* *yy* *x0* *y0*  
Create a transform. Without options, it is an identity transform. Given four arguments *xx*, *yx*, *xy*, and *yy*, it is a linear transform, given six arguments (with *x0* and *y0* last), it is an affine transform. Transforms can be called as functions on other transforms (concatenating them) or on points given either as complex number or real number pair. See also ly:make-rotation, ly:make-scaling, and ly:make-translation.

**ly:make-translation** *x* *y*  
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.
**ly:make-undead object**
This packages object in a manner that keeps it from triggering "Parsed object should be dead" messages.

**ly:make-unpure-pure-container unpure pure**
Make an unpure-pure container. unpure should be an unpure expression, and pure should be a pure expression. If pure is omitted, the value of unpure will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.

**ly:message str rest**
A Scheme callable function to issue the message str. The message is formatted with format and rest.

**ly:minimal-breaking pb**
Break (pages and lines) the Paper_book object pb without looking for optimal spacing: stack as many lines on a page before moving to the next one.

**ly:mm num**
num mm.

**ly:module->alist mod**
Dump the contents of module mod as an alist.

**ly:module-copy dest src**
Copy all bindings from module src into dest.

**ly:modules-lookup modules sym def**
Look up sym in the list modules, returning the first occurrence. If not found, return def or #f if def isn’t specified.

**ly:moment? x**
Is x a Moment object?

**ly:moment<? a b**
Compare two moments.

**ly:moment-add a b**
Add two moments.

**ly:moment-div a b**
Divide two moments.

**ly:moment-grace mom**
Extract grace timing as a rational number from mom.

**ly:moment-grace-denominator mom**
Extract denominator from grace timing.

**ly:moment-grace-numerator mom**
Extract numerator from grace timing.

**ly:moment-main mom**
Extract main timing as a rational number from mom.

**ly:moment-main-denominator mom**
Extract denominator from main timing.
Appendix A: Notation manual tables

ly:moment-main-numerator mom  
   Extract numerator from main timing.  

ly:moment-mod a b  
   Modulo of two moments.  

ly:moment-mul a b  
   Multiply two moments.  

ly:moment-sub a b  
   Subtract two moments.  

ly:music? obj  
   Is obj a music object?  

ly:music-compress m factor  
   Compress music object m by scale factor.  

ly:music-deep-copy m origin  
   Copy m and all sub expressions of m. m may be an arbitrary type; cons cells and music are copied recursively. If origin is given, it is used as the origin for one level of music by calling ly:set-origin! on the copy.  

ly:music-duration-compress mus fact  
   Compress mus by factor fact, which is a Moment.  

ly:music-duration-length mus  
   Extract the duration field from mus and return the length.  

ly:music-function? x  
   Is x a Music_function object?  

ly:music-function-extract x  
   Return the Scheme function inside x.  

ly:music-function-signature x  
   Return the function signature inside x.  

ly:music-length mus  
   Get the length of music expression mus and return it as a Moment object.  

ly:music-list? lst  
   Is lst a list of music objects?  

ly:music-mutable-properties mus  
   Return an alist containing the mutable properties of mus. The immutable properties are not available, since they are constant and initialized by the make-music function.  

ly:music-output? x  
   Is x a Music_output object?  

ly:music-property mus sym val  
   Return the value for property sym of music expression mus. If no value is found, return val or '() if val is not specified.  

ly:music-set-property! mus sym val  
   Set property sym in music expression mus to val.
ly:music-transpose \( m p \)  
Transpose \( m \) such that central C is mapped to \( p \). Return \( m \).

ly:note-column-accidentals \( note-column \)  
Return the AccidentalPlacement grob from \( note-column \) if any, or SCM_EOL otherwise.

ly:note-column-dot-column \( note-column \)  
Return the DotColumn grob from \( note-column \) if any, or SCM_EOL otherwise.

ly:note-head::stem-attachment \( font-metric \) \( glyph-name \)  
Get attachment in \( font-metric \) for attaching a stem to notehead \( glyph-name \).

ly:number->string \( s \)  
Convert \( s \) to a string without generating many decimals.

ly:one-line-auto-height-breaking \( pb \)  
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.

ly:one-line-breaking \( pb \)  
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.

ly:one-page-breaking \( pb \)  
Put each score on a single page. The paper-height settings are modified so each score fits on one page, and the height of the page matches the height of the full score.

ly:optimal-breaking \( pb \)  
Optimally break (pages and lines) the Paper_book object \( pb \) to minimize badness in both vertical and horizontal spacing.

ly:option-usage \( port \)  
Print ly:set-option usage. Optional \( port \) argument for the destination defaults to current output port.

ly:otf->cff \( otf-file-name \) \( idx \)  
Convert the contents of an OTF file to a CFF file, returning it as a string. The optional \( idx \) argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of \( idx \) is 0.

ly:otf-font? \( font \)  
Is \( font \) an OpenType font?

ly:otf-font-glyph-info \( font \) \( glyph \)  
Given the font metric \( font \) of an OpenType font, return the information about named glyph \( glyph \) (a string).

ly:otf-font-table-data \( font \) \( tag \)  
Extract a table \( tag \) from \( font \). Return empty string for non-existent \( tag \).

ly:otf-glyph-count \( font \)  
Return the number of glyphs in \( font \).

ly:otf-glyph-list \( font \)  
Return a list of glyph names for \( font \).
Appendix A: Notation manual tables

ly:output-def? x
   Is x a Output_def object? [Function]

ly:output-def-clone def
   Clone output definition def. [Function]

ly:output-def-lookup def sym val
   Return the value of sym in output definition def (e.g., \paper). If no value is found, return val or '()' if val is undefined. [Function]

ly:output-def-parent def
   Return the parent output definition of def. [Function]

ly:output-def-scope def
   Return the variable scope inside def. [Function]

ly:output-def-set-variable! def sym val
   Set an output definition def variable sym to val. [Function]

ly:output-description output-def
   Return the description of translators in output-def. [Function]

ly:output-find-context-def output-def context-name
   Return an alist of all context defs (matching context-name if given) in output-def. [Function]

ly:output-formats
   Formats passed to --format as a list of strings, used for the output. [Function]

ly:outputter-close outputter
   Close port of outputter. [Function]

ly:outputter-dump-stencil outputter stencil
   Dump stencil expr onto outputter. [Function]

ly:outputter-dump-string outputter str
   Dump str onto outputter. [Function]

ly:outputter-output-scheme outputter expr
   Output expr to the paper outputter. [Function]

ly:outputter-port outputter
   Return output port for outputter. [Function]

ly:page-marker? x
   Is x a Page_marker object? [Function]

ly:page-turn-breaking pb
   Optimally break (pages and lines) the Paper_book object pb such that page turns only happen in specified places, returning its pages. [Function]

ly:pango-font? f
   Is f a pango font? [Function]

ly:pango-font-physical-fonts f
   Return alist of (ps-name file-name font-index) lists for Pango font f. [Function]

ly:paper-book? x
   Is x a Paper_book object? [Function]
Appendix A: Notation manual tables

**ly:paper-book-header pb**
Return the header definition (\header) in Paper_book object pb.

**ly:paper-book-pages pb**

**ly:paper-book-paper pb**
Return the paper output definition (\paper) in Paper_book object pb.

**ly:paper-book-performances pb**

**ly:paper-book-scopes pb**

**ly:paper-book-systems pb**

**ly:paper-column::break-align-width col align-syms**
Determine the extent along the X-axis of a grob used for break-alignment organized by column col. The grob is specified by align-syms, which contains either a single break-align-symbol or a list of such symbols.

**ly:paper-column::print**
Optional stencil for PaperColumn or NonMusicalPaperColumn. Draws the rank number of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.

**ly:paper-fonts def**
Return a list containing the fonts from output definition def (e.g., \paper).

**ly:paper-get-font def chain**
Find a font metric in output definition def satisfying the font-qualifiers in alist chain chain, and return it. (An alist chain is a list of alists, containing grob properties.)

**ly:paper-get-number def sym**
Return the value of variable sym in output definition def as a double.

**ly:paper-outputscale def**
Return the output-scale for output definition def.

**ly:paper-score-paper-systems paper-score**
Return vector of paper_system objects from paper-score.

**ly:paper-system? obj**
Is obj a C++ Prob object of type paper-system?

**ly:paper-system-minimum-distance sys1 sys2**
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

**ly:parse-file name**

**ly:parse-string-expression parser-smob ly-code filename line**
Parse the string ly-code with parser-smob. Return the contained music expression. filename and line are optional source indicators.
**ly:parsed-undead-list!**
Return the list of objects that have been found live that should have been dead, and clear that list.

**ly:parser-clear-error parser**
Clear error flag for parser, defaulting to current parser.

**ly:parser-clone closures location**
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**
Bind symbol to val in current parser’s module.

**ly:parser-error msg input**
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.

**ly:parser-has-error? parser**
Does parser (defaulting to current parser) have an error flag?

**ly:parser-include-string ly-code**
Include the string ly-code into the input stream for current parser. Can only be used in immediate Scheme expressions ($ instead of #).

**ly:parser-lexer parser**
Return the lexer for parser, defaulting to current parser

**ly:parser-lookup symbol**
Look up symbol in current parser’s module. Return '() if not defined.

**ly:parser-output-name parser**
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**
Parse the string ly-code with parser-smob. Upon failure, throw ly-file-failed key.

**ly:parser-set-note-names names**
Replace current note names in parser. names is an alist of symbols. This only has effect if the current mode is notes.

**ly:performance-headers performance**
Return the list of headers with the innermost first.

**ly:performance-write performance filename name**
Write performance to filename storing name as the name of the performance in the file metadata.

**ly:pitch? x**
Is x a Pitch object?

**ly:pitch<? p1 p2**
Is p1 lexicographically smaller than p2?

**ly:pitch-alteration pp**
Extract the alteration from pitch pp.
Appendix A: Notation manual tables

ly:pitch-diff pitch root
Return pitch delta such that root transposed by delta equals pitch.

ly:pitch-negate p
Negate p.

ly:pitch-notename pp
Extract the note name from pitch pp.

ly:pitch-octave pp
Extract the octave from pitch pp.

ly:pitch-quartertones pp
Calculate the number of quarter tones of pp from middle C.

ly:pitch-semitones pp
Calculate the number of semitones of pp from middle C.

ly:pitch-steps p
Number of steps counted from middle C of the pitch p.

ly:pitch-tones pp
Calculate the number of tones of pp from middle C as a rational number.

ly:pitch-transpose p delta
Transpose p by the amount delta, where delta is relative to middle C.

ly:pointer-group-interface::add-grob grob sym grob-element
Add grob-element to grob’s sym grob array.

ly:position-on-line? sg spos
Return whether spos is on a line of the staff associated with the grob sg (even on an extender line).

ly:prob? x
Is x a Prob object?

ly:prob-immutable-properties prob
Retrieve an alist of immutable properties.

ly:prob-mutable-properties prob
Retrieve an alist of mutable properties.

ly:prob-property prob sym val
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
Is boolean prop sym of sym set?

ly:prob-set-property! obj sym value
Set property sym of obj to value.

ly:prob-type? obj type
Is obj the specified prob-type?

ly:programming-error str rest
A Scheme callable function to issue the internal warning str. The message is formatted with format and rest.
ly:progress str rest [Function]
A Scheme callable function to print progress str. The message is formatted with format and rest.

ly:property-lookup-stats sym [Function]
Return hash table with a property access corresponding to sym. Choices are prob, grob, and context.

ly:protects [Function]
Return hash of protected objects.

ly:pt num [Function]
num printer points.

ly:pure-call data grob start end rest [Function]
Convert property data (unpure-pure container or procedure) to value in a pure context defined by grob, start, end, and possibly rest arguments.

ly:randomize-rand-seed [Function]
Randomize C random generator.

ly:register-stencil-expression symbol [Function]
Add symbol as head of a stencil expression.

ly:register-translator creator name description [Function]
Register a translator creator (usually a descriptive alist or a function/closure returning one when given a context argument) with the given symbol name and the given description alist.

ly:relative-group-extent elements common axis [Function]
Determine the extent of elements relative to common in the axis direction.

ly:rename-file oldname newname [Function]
Rename oldname to newname. In contrast to Guile’s rename-file, 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:round-filled-box xext yext blot [Function]
Make a Stencil object that prints a black box of dimensions xext, yext and roundness blot.

ly:round-filled-polygon points blot extroversion [Function]
Make a Stencil object that prints a black polygon with corners at the points defined by points (list of coordinate pairs) and roundness blot. Optional extroversion shifts the outline outward, with the default of -1.0 keeping the outer boundary of the outline just inside of the polygon.

ly:run-translator mus output-def [Function]
Process mus according to output-def. An interpretation context is set up, and mus is interpreted with it. The context is returned in its final state.
 Optionally, this routine takes an object-key to to uniquely identify the score block containing it.

ly:score? x [Function]
Is x a Score object?
Appendix A: Notation manual tables

ly:score-add-output-def! score def
Add an output definition def to score.

ly:score-embedded-format score layout
Run score through layout (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

ly:score-error? score
Was there an error in the score?

ly:score-header score
Return score header.

ly:score-music score
Return score music.

ly:score-output-defs score
All output definitions in a score.

ly:score-set-header! score module
Set the score header.

ly:separation-item::print
Optional stencil for PaperColumn or NonMusicalPaperColumn. Draws the horizontal-skylines of each PaperColumn, showing the shapes used to determine the minimum distances between PaperColumns at the note-spacing step, before staves have been spaced (vertically) on the page.

ly:set-default-scale scale
Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

ly:set-grob-modification-callback cb
Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property.

ly:set-middle-C! context
Set the middleCPosition variable in context based on the variables middleCClefPosition and middleCOffset.

ly:set-option var val
Set a program option.

ly:set-origin! m origin
This sets the origin given in origin to m. m will typically be a music expression or a list of music. List structures are searched recursively, but recursion stops at the changed music expressions themselves. origin is generally of type ly:input-location?, defaulting to (*location*). Other valid values for origin are a music expression which is then used as the source of location information, or #f or '() in which case no action is performed. The return value is m itself.
ly:set-property-cache-callback cb  [Function]
Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property.

ly:skyline? x  [Function]
Is x a Skyline object?

ly:skyline-empty? sky  [Function]
Return whether sky is empty.

ly:skyline-pair? x  [Function]
Is x a Skyline_pair object?

ly:slur-score-count  [Function]
count number of slur scores.

ly:smob-protects  [Function]
Return LilyPond’s internal smob protection list.

ly:solve-spring-rod-problem springs rods length ragged  [Function]
Solve a spring and rod problem for count objects, that are connected by count-1 springs, and an arbitrary number of rods. count is implicitly given by springs and rods. The springs argument has the format (ideal, inverse_hook) and rods is of the form (idx1, idx2, distance).

length is a number, ragged a boolean.
The function returns a list containing the force (positive for stretching, negative for compressing and #f for non-satisfied constraints) followed by spring-count+1 positions of the objects.

ly:source-file? x  [Function]
Is x a Source_file object?

ly:source-files parser-smob  [Function]
A list of input files that have been opened up to here, including the files that have been closed already. a PARSER may optionally be specified.

ly:spanner? g  [Function]
Is g a spanner object?

ly:spanner-bound spanner dir  [Function]
Get one of the bounds of spanner. dir is -1 for left, and 1 for right.

ly:spanner-broken-into spanner  [Function]
Return broken-into list for spanner.

ly:spanner-set-bound! spanner dir item  [Function]
Set grob item as bound in direction dir for spanner.

ly:spawn command rest  [Function]
Simple interface to g_spawn_sync str. The error is formatted with format and rest.

ly:spring? x  [Function]
Is x a Spring object?
**ly:spring-set-inverse-compress-strength!**  
*spring strength*  
Set the inverse compress *strength* of *spring*.

**ly:spring-set-inverse-stretch-strength!**  
*spring strength*  
Set the inverse stretch *strength* of *spring*.

**ly:staff-symbol-line-thickness**  
*grob*  
Returns the current staff-line thickness in the staff associated with *grob*, expressed as a multiple of the current staff-space height.

**ly:staff-symbol-staff-radius**  
*grob*  
Returns the radius of the staff associated with *grob*.

**ly:staff-symbol-staff-space**  
*grob*  
Returns the current staff-space height in the staff associated with *grob*, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

**ly:start-environment**  
Return the environment (a list of strings) that was in effect at program start.

**ly:stderr-redirect**  
*file-name*  
*mode*  
Redirect stderr to *file-name*, opened with *mode*.

**ly:stencil?**  
*x*  
Is *x* a Stencil object?

**ly:stencil-add**  
*args*  
Combine stencils. Takes any number of arguments.

**ly:stencil-aligned-to**  
*stil*  
*axis dir*  
Align *stil* using its own extents. *dir* is a number. -1 and 1 are left and right, respectively. Other values are interpolated (so 0 means the center).

**ly:stencil-combine-at-edge**  
*first axis direction second padding*  
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be ’ () or #f.

**ly:stencil-empty?**  
*stil*  
*axis*  
Return whether *stil* is empty. If an optional axis is supplied, the emptiness check is restricted to that axis.

**ly:stencil-expr**  
*stil*  
Return the expression of *stil*.

**ly:stencil-extent**  
*stil*  
*axis*  
Return a pair of numbers signifying the extent of *stil* in *axis* direction (0 or 1 for x and y axis, respectively).

**ly:stencil-in-color**  
*stc r g b a*  
Put *stc* in a different color. Accepts either three values for *r*, *g*, *b* and an optional value for *a*, or a single CSS-like string.

**ly:stencil-outline**  
*stil outline*  
Return a stencil with the stencil expression (inking) of stencil *stil* but with outline and dimensions from stencil *outline*. 
ly:stencil-rotate stil angle x y  [Function]  
Return a stencil stil rotated angle degrees around the relative offset (x, y). E.g., an offset of (-1, 1) will rotate the stencil around the left upper corner.

ly:stencil-rotate-absolute stil angle x y  [Function]  
Return a stencil stil rotated angle degrees around point (x, y), given in absolute coordinates.

ly:stencil-scale stil x y  [Function]  
Scale stencil stil using the horizontal and vertical scaling factors x and y. Negative values will flip or mirror stil without changing its origin; this may result in collisions unless it is repositioned.

ly:stencil-stack first axis direction second padding mindist  [Function]  
Construct a stencil by stacking second next to first. axis can be 0 (x-axis) or 1 (y-axis). direction can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with padding as extra space. first and second may also be '()' or '#f'. As opposed to ly:stencil-combine-at-edge, metrics are suited for successively accumulating lines of stencils. Also, second stencil is drawn last.

If mindist is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, padding and mindist do not apply.

ly:stencil-translate stil offset  [Function]  
Return a stil, but translated by offset (a pair of numbers).

ly:stencil-translate-axis stil amount axis  [Function]  
Return a copy of stil but translated by amount in axis direction.

ly:stream-event? obj  [Function]  
Is obj a Stream_event object?

ly:string-percent-encode str  [Function]  
Encode all characters in string str with hexadecimal percent escape sequences, with the following exceptions: characters - , . /, and _; and characters in ranges 0-9, A-Z, and a-z.

ly:string-substitute a b s  [Function]  
Replace string a by string b in string s.

ly:system-font-load name  [Function]  
Load the OpenType system font name.otf. Fonts loaded with this command must contain three additional SFNT font tables called LILC, LILF, and LILY, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.

Note that only ly:font-get-glyph and derived code (like \lookup) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.

ly:text-interface::interpret-markup  [Function]  
Convert a text markup into a stencil. Takes three arguments, layout, props, and markup.
layout is a \layout block; it may be obtained from a grob with ly:grob-layout. props is an alist chain, i.e. a list of alists. This is typically obtained with (ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults)). markup is the markup text to be processed.

ly:transform? x  [Function]  
Is x a Transform object?
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ly:transform-&gt;list transform</code></td>
<td>Convert a transform matrix to a list of six values. Values are <code>xx</code>, <code>yx</code>, <code>xy</code>, <code>yy</code>, <code>x0</code>, <code>y0</code>.</td>
</tr>
<tr>
<td><code>ly:translate-cpp-warning-scheme str</code></td>
<td>Translates a string in C++ printf format and modifies it to use it for scheme formatting.</td>
</tr>
<tr>
<td><code>ly:translator? x</code></td>
<td>Is <code>x</code> a <code>Translator</code> object?</td>
</tr>
<tr>
<td><code>ly:translator-context trans</code></td>
<td>Return the context of the translator object <code>trans</code>.</td>
</tr>
<tr>
<td><code>ly:translator-description creator</code></td>
<td>Return an alist of properties of translator definition <code>creator</code>.</td>
</tr>
<tr>
<td><code>ly:translator-group? x</code></td>
<td>Is <code>x</code> a <code>Translator_group</code> object?</td>
</tr>
<tr>
<td><code>ly:translator-name creator</code></td>
<td>Return the type name of the translator definition <code>creator</code>. The name is a symbol.</td>
</tr>
<tr>
<td><code>ly:transpose-key-alist l pit</code></td>
<td>Make a new key alist of <code>l</code> transposed by pitch <code>pit</code>.</td>
</tr>
<tr>
<td><code>ly:truncate-list! lst i</code></td>
<td>Take at most the first <code>i</code> of list <code>lst</code>.</td>
</tr>
<tr>
<td><code>ly:ttf-&gt;pfa ttf-file-name idx</code></td>
<td>Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional <code>idx</code> argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of <code>idx</code> is 0.</td>
</tr>
<tr>
<td><code>ly:ttf-ps-name ttf-file-name idx</code></td>
<td>Extract the PostScript name from a TrueType font. The optional <code>idx</code> argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of <code>idx</code> is 0.</td>
</tr>
<tr>
<td><code>ly:type1-&gt;pfa type1-file-name</code></td>
<td>Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in PFA format, pass through it.</td>
</tr>
<tr>
<td><code>ly:undead? x</code></td>
<td>Is <code>x</code> a <code>Undead</code> object?</td>
</tr>
<tr>
<td><code>ly:unit</code></td>
<td>Return the unit used for lengths as a string.</td>
</tr>
<tr>
<td><code>ly:unpure-call data grob rest</code></td>
<td>Convert property <code>data</code> (unpure-pure container or procedure) to value in an unpure context defined by <code>grob</code> and possibly <code>rest</code> arguments.</td>
</tr>
<tr>
<td><code>ly:unpure-pure-container? x</code></td>
<td>Is <code>x</code> a <code>Unpure_pure_container</code> object?</td>
</tr>
<tr>
<td><code>ly:unpure-pure-container-pure-part pc</code></td>
<td>Return the pure part of <code>pc</code>.</td>
</tr>
</tbody>
</table>
**ly:unpure-pure-container-unpure-part** \( pc \)  
Return the unpure part of \( pc \).

**ly:usage**  
Print usage message.

**ly:verbose-output?**  
Was verbose output requested, i.e. loglevel at least `DEBUG`?

**ly:version**  
Return the current lilypond version as a list, e.g., \((1 \ 3 \ 127 \ uu1)\).

**ly:warning** \( str \) \( rest \)  
A Scheme callable function to issue the warning \( str \). The message is formatted with `format` and \( rest \).

**ly:warning-located** \( location \) \( str \) \( rest \)  
A Scheme callable function to issue the warning \( str \) at the specified location in an input file. The message is formatted with `format` and \( rest \).

**ly:wide-char->utf-8** \( wc \)  
Encode the Unicode codepoint \( wc \), an integer, as UTF-8.
Appendix B Cheat sheet

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 8 16</td>
<td>durations</td>
<td><img src="image1.png" alt="Example" /></td>
</tr>
<tr>
<td>c4. c4..</td>
<td>augmentation dots</td>
<td><img src="image2.png" alt="Example" /></td>
</tr>
<tr>
<td>c d e f g a b</td>
<td>scale</td>
<td><img src="image3.png" alt="Example" /></td>
</tr>
<tr>
<td>fis bes</td>
<td>alteration</td>
<td><img src="image4.png" alt="Example" /></td>
</tr>
<tr>
<td>\clef treble \clef bass</td>
<td>clefs</td>
<td><img src="image5.png" alt="Example" /></td>
</tr>
<tr>
<td>\time 3/4 \time 4/4</td>
<td>time signature</td>
<td><img src="image6.png" alt="Example" /></td>
</tr>
<tr>
<td>r4 r8</td>
<td>rest</td>
<td><img src="image7.png" alt="Example" /></td>
</tr>
<tr>
<td>d ~ d</td>
<td>tie</td>
<td><img src="image8.png" alt="Example" /></td>
</tr>
<tr>
<td>\key es \major</td>
<td>key signature</td>
<td><img src="image9.png" alt="Example" /></td>
</tr>
</tbody>
</table>
Note' raise octave

Note, lower octave

c(d e) slur

c\(c(d)e\) phrasing slur

a8[ b] beam

<< \new Staff ... >> more staves

c-> c-. articulations

c2\mf c\sfz dynamics

a\< a a\! crescendo
a\> a a\!

decrescendo

< >

chord

\partial 8

pickup / upbeat

\tuplet 3/2 \{f g a\}

triplets

\grace

grace notes

\lyricmode \{ twinkle \}

entering lyrics
twinkle

\new Lyrics

printing lyrics
twinkle

twin -- kle

lyric hyphen
twin - kle

\chordmode \{ c:\dim f:maj7 \}

chords

\new ChordNames

printing chord names

C° F°

<<\{e f\} \ \{c d\}>>

polyphony

s4 s8 s16

spacer rests
Appendix C GNU Free Documentation License

Version 1.3, 3 November 2008
https://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both
covers must also clearly and legibly identify you as the publisher of these copies. The front
cover must present the full title with all words of the title equally prominent and visible.
You may add other material on the covers in addition. Copying with changes limited to
the covers, as long as they preserve the title of the Document and satisfy these conditions,
can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the
first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto
adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you
must either include a machine-readable Transparent copy along with each Opaque copy,
or state in or with each Opaque copy a computer-network location from which the general
network-using public has access to download using public-standard network protocols a
complete Transparent copy of the Document, free of added material. If you use the latter
option, you must take reasonably prudent steps, when you begin distribution of Opaque
copies in quantity, to ensure that this Transparent copy will remain thus accessible at the
stated location until at least one year after the last time you distribute an Opaque copy
(directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before
redistributing any large number of copies, to give them a chance to provide you with an
updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions
of sections 2 and 3 above, provided that you release the Modified Version under precisely
this License, with the Modified Version filling the role of the Document, thus licensing
distribution and modification of the Modified Version to whoever possesses a copy of it. In
addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the
   Document, and from those of previous versions (which should, if there were any, be
   listed in the History section of the Document). You may use the same title as a previous
   version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for
   authorship of the modifications in the Modified Version, together with at least five of
   the principal authors of the Document (all of its principal authors, if it has fewer than
   five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the
   publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copy-
   right notices.

F. Include, immediately after the copyright notices, a license notice giving the public
   permission to use the Modified Version under the terms of this License, in the form
   shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover
   Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating
   at least the title, year, new authors, and publisher of the Modified Version as given
   on the Title Page. If there is no section Entitled “History” in the Document, create
   one stating the title, year, authors, and publisher of the Document as given on its
Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled “History” in the various original
documents, forming one section Entitled “History”; likewise combine any sections Entitled
“Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections
Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS
You may make a collection consisting of the Document and other documents released under
this License, and replace the individual copies of this License in the various documents with
a single copy that is included in the collection, provided that you follow the rules of this
License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually
under this License, provided you insert a copy of this License into the extracted document,
and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS
A compilation of the Document or its derivatives with other separate and independent
documents or works, in or on a volume of a storage or distribution medium, is called an
“aggregate” if the copyright resulting from the compilation is not used to limit the legal
rights of the compilation’s users beyond what the individual works permit. When the
Document is included in an aggregate, this License does not apply to the other works in
the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document,
then if the Document is less than one half of the entire aggregate, the Document’s Cover
Texts may be placed on covers that bracket the Document within the aggregate, or the
electronic equivalent of covers if the Document is in electronic form. Otherwise they must
appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
Translation is considered a kind of modification, so you may distribute translations of the
Document under the terms of section 4. Replacing Invariant Sections with translations
requires special permission from their copyright holders, but you may include translations
of some or all Invariant Sections in addition to the original versions of these Invariant
Sections. You may include a translation of this License, and all the license notices in
the Document, and any Warranty Disclaimers, provided that you also include the original
English version of this License and the original versions of those notices and disclaimers. In
case of a disagreement between the translation and the original version of this License or a
notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”,
the requirement (section 4) to Preserve its Title (section 1) will typically require changing
the actual title.

9. TERMINATION
You may not copy, modify, sublicense, or distribute the Document except as expressly pro-
vided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute
it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copy-
right holder is reinstated (a) provisionally, unless and until the copyright holder explicitly
and finally terminates your license, and (b) permanently, if the copyright holder fails to
notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the
copyright holder notifies you of the violation by some reasonable means, this is the first
time you have received notice of violation of this License (for any work) from that copyright
holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.
ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ``GNU Free Documentation License''.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with... Texts." line with this:

    with the Invariant Sections being list their titles, with
    the Front-Cover Texts being list, and with the Back-Cover Texts
    being list.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.
Appendix D  LilyPond command index

This index lists all the LilyPond commands and keywords with links to those sections of the manual that describe or discuss their use.

! .................................................. 6
\! .................................................. 125

" .................................................. 112

% 
% .................................................. 487, 491
%{ ... %} ........................................ 487, 491

, .................................................. 1

( 
( .................................................. 133
\( .................................................. 136

) 
) .................................................. 133
\) .................................................. 136

, .................................................. 1

_ 
_ .................................................. 122, 444
_! .................................................. 768
_+ .................................................. 769
_-- .............................................. 768
_-- .............................................. 768
_-- .............................................. 768
_-- .............................................. 768
_-- .............................................. 768
_-- .............................................. 768

. .................................................. 48

/ 
/ .................................................. 428
/\ .................................................. 428

: .................................................. 166

< 
< .................................................. 168
\< .................................................. 125
<...> ........................................... 168
<> .............................................. 169, 340

= 
= .................................................. 10
\= .............................................. 134, 821

> 
> .................................................. 168
\> .............................................. 125

? 
? .................................................. 6

[ 
[ .................................................. 96
\[ .............................................. 448

] 
] .................................................. 96
\] .............................................. 448

^ 
^ .................................................. 427

_ 
_ .................................................. 282

| 
| .................................................. 112

~ 
~ .................................................. 55

-
Appendix D: LilyPond command index

\cadenzaff \cadenzaf \cesura \cleft \center-align \center-column \change \char \check-consistency \choral \choral-cautionary \chordChanges \chordmode \chordNameExceptions \chordNameLowerCaseMinor \chordNames \chordNameSeparator \chordNoteNamer \chordPrefixSpacer \chordRepeats \chordRootNamer \chords \circle \circuslus \clef \clip-regions \cm \coda \color \column \column-lines \combine \common-shortest-duration \Completion_heads_ engraver \Completion_rest_engraver \compound-meter \compoundMeter \compressEmptyMeasures \compressMMRests \concat \consists \context \context-spec-music \controlpitch \countPercentRepeats \cr \cresc \crescendo-event \crescendoSpanner \crescendoText \crescTextCresc \cross \crossStaff \cueClef \cueClefUnset \cueDuring \cueDuringWithClef \cueVoice \currentBarNumber \customTabClef

\dashBang \dashDash \dashDot \dashHat \dashLarger \dashPlus \dashUnderscore \deadNote \deadNotesOff \deadNotesOn \decr \decresc \decrescendoSpanner \decrescendoText

default \default \default-staff-staff-spacing \defaultBarType \defaultChild \defaultTimeSignature \defineBarLine \deminutum \dennes \descends \dim \dimHairpin \dimTextDecr \dimTextDecresc \dimTextDim \dir-column \disant \displayLilyMusic \displayMusic \displayScheme \divisioMaior \divisioMaxima \divisioMinima \dodecaphonic \dodecaphonic-first \dodecaphonic-no-repeat \dorian 
dotsDown 
dotsNeutral 
dotsUp 
doubleflat 
doublesharp 
doublestroke 
douv
doomdort 
downmordent 
downprall 
downrall 
draw-circle 
draw-dashed-line 
draw-dotted-line 
draw-hline 
draw-line 
draw-squiggly-line 
dropNote 
drummode 
drumPitchNames 
drumPitchTable 
drums 
DrumStaff 
drumStyleTable
Appendix D: LilyPond command index

S

S................. 61
\\sacredHarpsHeads.................... 42
\\sacredHarpsHeadsMinor.............. 43
\\sas................. 748
\\scale.................. 743
\\scaleDurations................. 54, 77, 818
\\score................... 486, 490, 747
\\core-lines.................. 765
score-markup-spacing.............. 550
score-system-spacing............. 551
scoreTitleMarkup.................. 500
\\segno.................. 122, 770
self-alignment-X.................. 567
\\semicircle.............. 460, 770
\\semiflat............... 748
\\semigermanChords............. 432
\\semisharp............... 749
\\sesquiflat.............. 749
\\sesquisharp............. 749
\\set................... 87, 620, 625
set-global-fonts................. 272
set-global-staff-size............ 559
\\settingsFrom............. 818
\\sf.................... 125
\\sfz..................... 125
\\shape................. 656, 818
\\sharp................... 749
\\shiftDurations............. 819
\\shiftOff................. 178
\\shiftOn................. 178
\\shiftOnew............ 178
\\newName............... 583
short-indent................... 211, 553
\\shortfermata............. 122, 769
show-available-fonts............. 12, 271
showFirstLength................. 528, 775
\\showKeySignature............. 413
showLastLength................. 528, 775
\\showStaffSwitch............. 344
\\signaturecongruentiae.... 122, 770
\\simple.................. 717
\\single................. 508, 628, 819
\\skip.................... 61, 299, 819
skipBars........................ 223
skipTypesetting.................. 528
slashChordSeparator.............. 433
\\slashed-digit............ 762
\\slashedGrace.............. 115, 819
\\slashSeparator............. 556
\\slashturn.................. 122, 768
\\slur-event................. 216
\\slurDashed................. 134
\\slurDashPattern............. 135, 819
\\slurDotted................. 134
\\slurDown.................. 134
\\slurHalfDashed.............. 134
\\slurHalfSolid.............. 134
\\slurNeutral................. 134
\\slurSolid.................. 134
\\slurUp..................... 135
\\small................. 225, 256, 717
\\smallCaps.................. 717
\\smaller.................. 254, 256, 717
\\smappizzicato................. 122, 769
\\sostenutoOff.................. 347
\\sostenutoOn.................. 347
\\sourcefileline............. 491
\\sourcefilename.............. 491
\\southernHarmonyHeads........... 42
\\southernHarmonyHeadsMinor..... 43
\\sp........................ 125
\\spaning.................... 583
\\spacingTweaks.................. 819
\\Span_stem_ engraver............ 345
\\spft...................... 125
\\staccatissimo................. 122, 768
\\staccato.................. 122, 768
\\staff-affinity.............. 567
\\staff.padding.............. 230
\\staff-space................. 635
\\staff-staff-spacing........... 567
\\Staff.midrinstrument......... 542
\\Staff_collecting_ engraver... 250
\\Staff_symbol_ engraver........ 205
\\staffgroup-staff-spacing.... 567
\\start-repeat................. 159
\\startacciaccaturaMusic........ 118
\\startAppoggiaturaMusic........ 118
\\startGraceMusic.............. 118
\\startGroup.................. 240
\\startStaff................. 199, 202
\\startTrillSpan................ 148
\\standardBass................ 754
\\standardBassIV............... 755
\\standardBassV................ 756
\\standardBassVI................. 757
\\stem-spacing-correction........ 583
\\stemDown...................... 234
\\stemLeftBeamCount.............. 96, 97
\\stemNeutral.................. 234
\\stemRightBeamCount............. 96, 97
\\stemUp....................... 234
\\stencil...................... 763
\\stopacciaccaturaMusic......... 118
\\stopAppoggiaturaMusic......... 118
\\stopGraceMusic.............. 118
\\stopGroup.................. 240
\\stopped................... 122, 407, 411, 769
\\stopStaff.................. 199, 202, 205
\\stopTrillSpan................ 148
\\storePredefinedDiagram........ 387, 392, 819
\\strictBeatBeaming............. 92
\\stringNumberorientations.... 230
\\stringTuning.................. 370, 819
\\stringTunings.................. 370, 383
\\strokeFingerorientations..... 230, 397
\\strpho...................... 461, 468
\\strut...................... 763
\\styledNoteHeads.............. 819
\\sub........................ 255, 718
\\subdivideBeams.............. 91
\\suggestAccidentals........... 124, 455
\\super...................... 255, 718
\\sus........................ 427
\\sustainOff................. 347
\\sustainOn................. 347
\\system-count................. 554
Appendix D: LilyPond command index

W
\walkerHeads ........................................... 42
\walkerHeadsMinor .................................... 43
\whichBar ............................................. 106
\whiteout ............................................ 763
\whiteTriangleMarkup .................................. 432
\with ................................................. 605, 610
\with-color .......................................... 232, 764
\with-dimensions .................................... 764
\with-dimensions-from ................................ 764
\with-link ........................................... 765
\with-url ............................................ 744
\withMusicProperty ................................... 821
\woodwind-diagram .................................... 752
\wordwrap ............................................ 261, 735
\wordwrap-field ...................................... 734
\wordwrap-internal ................................... 766
\wordwrap-lines ...................................... 267, 766
\wordwrap-string ..................................... 735
\wordwrap-string-internal ............................ 767

X
x11-color ............................................. 232, 233
X-offset ............................................... 567
\xNote ............................................... 39, 821
\xNotesOff .......................................... 39
\xNotesOn ........................................... 39
Appendix E  LilyPond index

In addition to all the LilyPond commands and keywords, this index lists musical terms and words that relate to each of them, with links to those sections of the manual that describe or discuss that topic.

Slanted index entries point to locations (mostly ‘See also’ sections) that contain external links to other LilyPond documentation files like the Internal Reference or the Glossary.
Appendix E: LilyPond index

1
15ma ........................................... 24
15mb ........................................... 24

8
8va ........................................... 24
8vb ........................................... 24

A
a due .......................................... 188
a due part ................................... 183
\abs-fontsize ................................ 254, 711
absolute ..................................... 1
\absolute .................................... 810
absolute dynamics .......................... 125
absolute octave entry ........................ 1
absolute octave specification .............. 1
accent ........................................ 125
accent ........................................ 123
\accent ....................................... 122, 768
\accentus .................................... 460, 770
'accentus' gregorian articulation ......... 460, 770
\accepts ..................................... 612, 613, 614
acciaccatura ................................ 119
acciaccatura ................................ 115
\acciaccatura ................................ 115, 810
acciaccatura, multi-note .................... 119
Accidental .................................... 7, 35
accidental .................................... 455, 459, 470
accidental .................................... 6, 29
accidental style .............................. 29
accidental style, chordal .................... 32
accidental style, cautionary ............... 32
accidental style, default .................... 29, 30
accidental style, dodecaphonic .......... 34
accidental style, dodecaphonic-first .... 34
accidental style, dodecaphonic-no-repeat 34
accidental style, forget ..................... 35
accidental style, modern ................... 30, 31
accidental style, modern-cautionary ...... 30, 31
accidental style, modern-voice-cautionary 32
accidental style, neo-modern .............. 33
accidental style, neo-modern-cautionary ... 33
accidental style, neo-modern-voice ....... 33
accidental style, neo-modern-voice-cautionary 33
accidental style, no-reset .................. 34
accidental style, piano ..................... 32
accidental style, piano-cautionary ....... 32
accidental style, teaching .................. 34
accidental style, voice ..................... 30
accidental, and simultaneous notes ....... 35
accidental, automatic ....................... 29
accidental, cautionary ...................... 6
accidental, Gregorian ...................... 459
accidental, hiding, on tied notes at start of system .................. 7
accidental, in cadenzas .................... 75
accidental, in chords ....................... 35
accidental, in unmetred music ............. 75
accidental, Kievian ......................... 470
accidental, mensural ....................... 455
accidental, modern style ................... 31
accidental, multi-voice ..................... 31
accidental, musica ficta .................... 455
accidental, on tied note .................... 6
accidental, parenthesized ................... 6
accidental, quarter-tone .................... 7
accidental, reminder ....................... 6
accidental-interface ....................... 7
accidental-suggestion-interface ......... 35
Accidental_engraver ....................... 7, 33, 456
AccidentalCautionary ...................... 7
AccidentalPlacement ...................... 35
\accidentalStyle ......................... 29, 810
AccidentalSuggestion ...................... 35, 456
\accent ....................................... 124
accordion .................................... 348
accordion discord symbol .................. 348
accordion register symbol ................ 348
accordion shift symbol .................... 348
acoustic bass ................................ 770
acoustic snare ................................ 770
add-grace-property ........................ 117
add-stem-support ............................ 230
add-toc-item! ................................ 516
addChordShape ................................ 387, 810
Adding and removing engravers ............ 81
Adding custom fret diagram ............... 386
adding white background, to text .......... 763
\addInstrumentDefinition .................. 810
addition, in chords ......................... 427
additional voices in polyphonic music ... 181
additionalPitchPrefix ..................... 432
\addlyrics ................................ 276, 278, 279
\addQuote ................................... 214, 810
adjusting staff symbol .................... 636
Advanced command line options for LilyPond 530
aeolian ....................................... 22
\aeolian ..................................... 22
\afterGrace .................................. 116, 810
\afterGraceFraction ........................ 775
agogo .......................................... 770
Aiken shape note head ..................... 42
Aiken shape note head, thin variant ..... 44
\aikenHeads ................................ 42
\aikenHeadsMinor ......................... 43
\aikenThinHeads ......................... 42
\aikenThinHeadsMinor .................... 43
al niente ................................... 131
al niente, hairpin ........................... 129
\alias ......................................... 612
align to object ................................ 652
alignBelowContext ......................... 204, 616
Aligning lyrics to a melody .............. 278, 278, 286
aligning markup ............................. 257
aligning markup text ........................ 257
aligning text ................................ 257
aligning to cadenza ......................... 120
alignment, bar numbers .................... 111
alignment, horizontal, lyrics ............. 294
alignment, text, commands ................. 261
alignment, vertical, dynamics ............. 130
alignment, vertical, text ................... 258
alignment, vertical, text scripts ......... 130
alist ........................................... 773
\autoLineBreaksOn ............................................. 560
automatic accidental .................................... 29
automatic chord diagram .................................. 393
automatic fret diagram .................................... 393
automatic part combining .................................. 183
automatic staff change .................................... 342
automaticBars ............................................. 647
\autoPageBreaksOff ......................................... 563
\autoPageBreaksOn .......................................... 563
available fonts, listing ..................................... 271
Axis_group _engraver ...................................... 210, 572

B

Bézier curve, control points ............................. 655
Backend ...................................................... 617, 619, 623
backslashed digit .......................................... 758
\backslashed-digit ......................................... 758
bagpipe ...................................................... 413
bagpipe example .......................................... 414
balance in MIDI ............................................ 541
balloon ....................................................... 237
balloon help ................................................ 237
balloon-interface .......................................... 238
Balloon_engraver ......................................... 238
Balloon_engraver ......................................... 237
\balloonGrobText ......................................... 237, 811
\balloonLengthOff ......................................... 237
\balloonLengthOn ......................................... 237
\balloonText ................................................ 237, 811
BalloonTextEdit ........................................... 238
banjo tablature ............................................ 354, 401
banjo tuning ................................................ 401
banjo-c-tuning ............................................. 401
banjo-modal-tuning ......................................... 401
banjo-open-d-tuning ......................................... 401
banjo-open-dm-tuning ......................................... 401
\bar .......................................................... 99, 106, 811
bar check ...................................................... 112
bar check, with repeats ................................... 153
bar line ....................................................... 99
bar line, avoided by lyrics ................................ 295
bar line, between staves .................................. 104, 195
bar line, cadenzas ......................................... 75
bar line, closing ............................................ 99
bar line, default, changing ................................ 106
bar line, defining .......................................... 104
bar line, double ............................................ 99
bar line, hairpin, stopping at ............................ 128
bar line, in ChordNames ................................... 435
bar line, invisible ......................................... 100
bar line, manual ............................................ 100
bar line, suppressing ...................................... 647
bar line, symbols on ....................................... 248
bar line, unmetered music ................................ 75
bar number ................................................... 107, 121
bar number check .......................................... 112
bar number, alignment .................................... 111
bar number, alternative, in repeat ...................... 157
bar number, cadenzas ...................................... 75
bar number, collision ...................................... 112
bar number, printed at regular intervals .............. 108
bar number, printed in first measure ................... 108
bar number, regular spacing ............................. 107
bar number, removal ....................................... 111
bar number, style .......................................... 109
bar number, unmetered music ......................... 75
bar number, with letter, in repeat ...................... 157
Bar_engraver ............................................. 431
Bar_number_engraver ...................................... 112
carCheckSynchronize ....................................... 112
carftNote .................................................... 17, 706
cartine varC clef ........................................... 706
cartine varF clef ........................................... 706
BarLine ...................................................... 107
BarNumber .................................................. 112
\barNumberCheck .......................................... 112, 811
barNumberVisibility ....................................... 107
basestandard-duration ................................. 583
baseMoment ............................................... 87, 91
Basic command line options for LilyPond ............ 528
bass .......................................................... 770
bass clef ..................................................... 17, 706
bass note, for chords ...................................... 428
bass, figured ............................................... 437
bass, thorough ............................................. 437
BassFigure .................................................. 441, 442
BassFigureAlignment ....................................... 441, 442
BassFigureBracket ......................................... 441, 442
BassFigureContinuation ................................... 441, 442
\bassFigureExtendersOff ................................... 439
\bassFigureExtendersOn ................................... 439
BassFigureLine ............................................ 441, 442
\bassFigureStaffAlignmentDown ......................... 442
\bassFigureStaffAlignmentNeutral ....................... 442
\bassFigureStaffAlignmentUp ............................ 442
basso continuo ............................................ 437
bayati ....................................................... 482
Beam ........................................................ 87, 94, 98, 342, 386
\beam ........................................................ 736
beam, cadenzas ............................................ 75
beam, cross-staff ......................................... 340
beam, customizing rules .................................. 84
beam, endings, in a score ................................ 93
beam, endings, with multiple voices .................... 93
beam, feathered ............................................ 98
beam, horizontal .......................................... 362
beam, in tablature ........................................ 362
beam, line breaks ......................................... 85
beam, manual .............................................. 84, 96
beam, niks ................................................... 97
beam, subdividing ......................................... 91
beam, unmetered music ................................... 75
beam, with knee gap ....................................... 85
beam, with knee gap, changing ......................... 85
beam, with lyrics .......................................... 87
beam, with melisma ........................................ 85
beam, with polymeric meters ............................. 77
beam-interface ........................................... 87, 94, 98
Beam_engraver ............................................ 87, 98
beamed tuplet, line break within ....................... 53
BeamEvent .................................................. 87, 98
\beamExceptions .......................................... 87, 811
BeamForbidEvent .......................................... 87, 94
Appendix E: LilyPond index

877

\cr ......................................................... 125

\crash_cymbal ............................................. 125

creating a table ........................................... 765

creating empty text object ............................. 761

creating horizontal space, in text .................... 726

creating text fraction .................................. 759

creating vertical space, in text ....................... 733, 763

\cresc ..................................................... 126

crescendo ................................................ 131

crescendo ................................................ 125

crescendo-event .......................................... 216

crescendospanner ........................................ 130

crescendotext ............................................ 130

\crescHairpin ............................................ 127

\crescTextCresc ......................................... 127

cross ..................................................... 39

cross note head ........................................... 39

cross-staff arpeggio .................................... 146

cross-staff arpeggio bracket .......................... 346

cross-staff beam ......................................... 340

cross-staff chord ........................................ 345

cross-staff collision .................................... 341

cross-staff line ......................................... 344

cross-staff note ......................................... 340, 345

cross-staff parenthesis-style arpeggio ............... 148

cross-staff stem ......................................... 345

cross-staff tremolo ...................................... 166

cross-voice arpeggio .................................... 147

\crossStaff ............................................. 345, 812

css color codes .......................................... 232

cue clef .................................................. 17

cue note .................................................... 214, 217

cue note, formatting ..................................... 217

cue note, removing ....................................... 221

cue, in vocal score ...................................... 322

cue, musical ............................................... 320

cue-notes .................................................. 324

\cueClef ................................................. 217, 812

\cueClefUnset ........................................... 217, 812

\cueDuring ............................................... 217, 812

\cueDuringWithClef ..................................... 217, 812

\cueVoice ................................................. 222, 324

\cueVoice ................................................. 217

cuica ..................................................... 770

customBarNumber ......................................... 107, 121

customes .................................................. 449

custom fret diagram .................................... 373, 382

custom fret diagram, adding .......................... 386

custom fretboard fret diagram ......................... 388

custom metronome mark markup ....................... 73

custom rehearsal mark .................................. 114

custom string tuning .................................... 370

customized fret diagram ................................ 381

customizing chord name ................................ 432

customTabClef ........................................... 744

custos ...................................................... 447, 450

\custos .................................................... 450

custos ...................................................... 449

cymbal, various ......................................... 770
Appendix E: LilyPond index

D

d.D.S. al Fine ........................................ 114
\dampened note, on fretted instrument ........ 398
\dash patterns, slur ......................... 135
\DashBang ........................................ 123
\DashDot .......................................... 123
\dashed phrasing slur ....................... 137
\dashed slur ................................ 134
\dashed tie ................................ 57
\DashHat ......................................... 123
\DashLarger ...................................... 123
\DashPlus ........................................ 123
\DashUnderscore .................................. 123
\dead note, percussion ...................... 409
\default ......................................... 39, 812
\defaultNotesOff ............................. 39
\defaultNotesOn ............................... 39
decorating text .............................. 262
\decr ........................................... 125
\darc ........................................... 126
decresc ......................................... 131
decrescando .................................... 125
decrescandoSpanner ............................. 130
decrescandoText ................................ 130
default ........................................ 29, 30
\default ........................................ 113, 506
default accidental style ........................ 29, 30
default bar line, changing ................... 106
default context property, changing ........ 607
default note duration ......................... 47
default note name ............................... 6
default pitch .................................. 47
default, offsetting ............................ 626
default-staff-staff-spacing .................. 567
\Default_bar_line_ engraver .................... 80
defaultBarType ................................ 106
\defaultCat ................................... 615
\defaultTimeSignature ......................... 67
define predefined fretboard .................. 389
\defineBarLine ................................ 104, 812
defining bar line ................................ 104
delayed turn ornamentation .................. 124
\ diminutum ...................................... 461, 468
\denies ......................................... 612, 614
\descends ....................................... 461, 468
desk parts ..................................... 210
diagram, chord for fretted instruments ...... 373
diagram, fret .................................... 373
diagram, fret, customized .................... 381
diagram, fret, transposing ................... 385
diamond note head ............................. 39
diamond-shaped note head ................... 352
Difficult tweaks ................................ 661
dim ................................................. 425
\dim ............................................. 126
dimension ....................................... 636
\dimHairpin ..................................... 127
diminuendo ....................................... 125
\dimTextDecr ................................... 127
\dimTextDecresc ................................ 127
\dimTextDim .................................... 127
\dir-column .................................... 722
direction, automatic, of stem, on center line . 235
direction, default, of stem, on center line . 235
\disant ........................................... 753
discant symbol, accordion .................. 348
\Displaying music expressions ................ 544, 625
\displayLilyMusic ................................ 544, 812
\displayMusic .................................... 812
\displayScheme .................................. 812
distance, absolute ............................ 635
distance, between staves ..................... 567
distance, scaled ................................ 635
divided lyrics .................................... 303
divided voices ................................... 317
divisi staves .................................... 208
divisi ............................................. 460
\divisio ......................................... 459
\divisioMajor .................................... 460
\divisioMinima ................................... 460
\divisioMaxima ................................... 460
divisiones ....................................... 459
dodecaphonic ................................... 34
dodecaphonic accidental style ............. 34
dodecaphonic-first ............................ 34
dodecaphonic-first accidental style ...... 34
dodecaphonic-no-repeat ....................... 34
dodecaphonic-no-repeat accidental style ... 34
dot ............................................... 140
dot ............................................... 140
dorian ........................................... 22
\dorian ......................................... 22
\DotColumn ...................................... 49
\Dots ............................................ 49
\DotsDown ....................................... 48
\dotsNeutral ..................................... 48
\dotsUp .......................................... 48
dotted note ..................................... 48
dotted note, change number of dots ......... 48
dotted note, moving horizontally ........... 181
dotted phrasing slur ........................... 137
dotted slur ..................................... 137
dotted tie ....................................... 57
double bar line .................................. 99
double flat ..................................... 7
double flat ..................................... 6
double sharp .................................... 7
double sharp .................................... 6
double slurs, for legato chords ............ 135
double-dotted note ............................. 48
double-time signature ....................... 77
\Double_percent_repeat_ engraver ........... 165
\doubleflat ..................................... 744
\DoublePercentEvent ......................... 165
\DoublePercentRepeat ......................... 165
\DoublePercentRepeatCounter ............... 165
\DoubleRepeatSlash ............................. 165
\doubleslash .................................... 745
doubleslurs ..................................... 135
\‘down pralltriller’ ornament .............. 122, 768
\‘down-bow’ bowing indication .......... 122, 769
down-bow indication ........................... 351
\downbow ........................................ 122, 351, 769
\downmordent .................................. 122, 768
\downprall ...................................... 122, 768
\draw ............................................ 263, 737
\draw-circle .................................... 737
\draw-dashed-line ............................. 738
\draw-dotted-line ............................. 738
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\draw-hline</td>
<td>738</td>
</tr>
<tr>
<td>\draw-line</td>
<td>263, 738</td>
</tr>
<tr>
<td>\draw-squiggle-line</td>
<td>739</td>
</tr>
<tr>
<td>drawing beam, within text</td>
<td>736</td>
</tr>
<tr>
<td>drawing box, with rounded corners</td>
<td>740</td>
</tr>
<tr>
<td>drawing box, with rounded corners, around text</td>
<td>743</td>
</tr>
<tr>
<td>drawing circle, within text</td>
<td>737</td>
</tr>
<tr>
<td>drawing dashed line, within text</td>
<td>737</td>
</tr>
<tr>
<td>drawing dotted line, within text</td>
<td>738</td>
</tr>
<tr>
<td>drawing ellipse, around text</td>
<td>739</td>
</tr>
<tr>
<td>drawing graphic object</td>
<td>262</td>
</tr>
<tr>
<td>drawing line, across a page</td>
<td>738</td>
</tr>
<tr>
<td>drawing line, within text</td>
<td>738</td>
</tr>
<tr>
<td>drawing oval, around text</td>
<td>740</td>
</tr>
<tr>
<td>drawing path</td>
<td>741</td>
</tr>
<tr>
<td>drawing solid box, within text</td>
<td>740</td>
</tr>
<tr>
<td>drawing squiggled line, within text</td>
<td>739</td>
</tr>
<tr>
<td>drawing staff symbol</td>
<td>636</td>
</tr>
<tr>
<td>drawing triangle, within text</td>
<td>743</td>
</tr>
<tr>
<td>\dropNote</td>
<td>429, 812</td>
</tr>
<tr>
<td>drum</td>
<td>403, 404</td>
</tr>
<tr>
<td>drum staff</td>
<td>191</td>
</tr>
<tr>
<td>drum, various</td>
<td>770</td>
</tr>
<tr>
<td>\drummode</td>
<td>191, 403, 633</td>
</tr>
<tr>
<td>drumPitchNames</td>
<td>407</td>
</tr>
<tr>
<td>drumPitchTable</td>
<td>408</td>
</tr>
<tr>
<td>\drums</td>
<td>403, 633</td>
</tr>
<tr>
<td>DrumStaff</td>
<td>192, 409</td>
</tr>
<tr>
<td>DrumStyleTable</td>
<td>197</td>
</tr>
<tr>
<td>DrumVoice</td>
<td>409</td>
</tr>
<tr>
<td>duration line</td>
<td>444</td>
</tr>
<tr>
<td>Duration names notes and rests</td>
<td>49</td>
</tr>
<tr>
<td>duration, default</td>
<td>47</td>
</tr>
<tr>
<td>duration, isolated</td>
<td>47</td>
</tr>
<tr>
<td>duration, of note</td>
<td>47</td>
</tr>
<tr>
<td>duration, scaling</td>
<td>54</td>
</tr>
<tr>
<td>\endline</td>
<td>480</td>
</tr>
<tr>
<td>\dynamic</td>
<td>131, 712</td>
</tr>
<tr>
<td>dynamic mark, multiple on one note</td>
<td>126</td>
</tr>
<tr>
<td>dynamic mark, new</td>
<td>131</td>
</tr>
<tr>
<td>dynamic-event</td>
<td>216</td>
</tr>
<tr>
<td>Dynamic_performer</td>
<td>594, 535, 537</td>
</tr>
<tr>
<td>\dynamicDown</td>
<td>127</td>
</tr>
<tr>
<td>DynamicLineSpanner</td>
<td>127, 131</td>
</tr>
<tr>
<td>DynamicLineSpanner</td>
<td>127, 130</td>
</tr>
<tr>
<td>\dynamicNeutral</td>
<td>127</td>
</tr>
<tr>
<td>Dynamics</td>
<td>131</td>
</tr>
<tr>
<td>dynamics</td>
<td>125</td>
</tr>
<tr>
<td>dynamics in MIDI</td>
<td>533</td>
</tr>
<tr>
<td>dynamics in MIDI, custom</td>
<td>533</td>
</tr>
<tr>
<td>dynamics, absolute</td>
<td>125</td>
</tr>
<tr>
<td>dynamics, centered in keyboard music</td>
<td>339</td>
</tr>
<tr>
<td>dynamics, editorial</td>
<td>131</td>
</tr>
<tr>
<td>dynamics, extender line, hiding</td>
<td>130</td>
</tr>
<tr>
<td>dynamics, parenthesis</td>
<td>131</td>
</tr>
<tr>
<td>dynamics, text spanner, custom</td>
<td>247</td>
</tr>
<tr>
<td>dynamics, text, style</td>
<td>130</td>
</tr>
<tr>
<td>dynamics, vertical alignment</td>
<td>130</td>
</tr>
<tr>
<td>dynamics, vertical positioning</td>
<td>127</td>
</tr>
<tr>
<td>DynamicText</td>
<td>131</td>
</tr>
<tr>
<td>\dynamicUp</td>
<td>127</td>
</tr>
</tbody>
</table>

**E**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>easy notation</td>
<td>40</td>
</tr>
<tr>
<td>easy notation, using numbers</td>
<td>40</td>
</tr>
<tr>
<td>easy play note head</td>
<td>40</td>
</tr>
<tr>
<td>\easyHeadsOff</td>
<td>192, 409</td>
</tr>
<tr>
<td>\emptyChord</td>
<td>118, 340</td>
</tr>
<tr>
<td>\emptyStaff</td>
<td>205</td>
</tr>
<tr>
<td>\epsil</td>
<td>275</td>
</tr>
<tr>
<td>\epsil</td>
<td>51</td>
</tr>
<tr>
<td>\epsil</td>
<td>461</td>
</tr>
<tr>
<td>\epsil</td>
<td>461</td>
</tr>
<tr>
<td>\epsil</td>
<td>461</td>
</tr>
<tr>
<td>\epsil</td>
<td>460</td>
</tr>
<tr>
<td>\epsil</td>
<td>460</td>
</tr>
<tr>
<td>\epsil</td>
<td>460</td>
</tr>
<tr>
<td>\epsil</td>
<td>528</td>
</tr>
<tr>
<td>\epsil</td>
<td>263, 740</td>
</tr>
<tr>
<td>\epsil</td>
<td>434</td>
</tr>
<tr>
<td>equalization, instrument, replacing MIDI default</td>
<td>535</td>
</tr>
<tr>
<td>expressive</td>
<td>122, 126, 768</td>
</tr>
<tr>
<td>'expressivo' articulation</td>
<td>122, 126, 768</td>
</tr>
<tr>
<td>\etc</td>
<td>664</td>
</tr>
<tr>
<td>\eventChords</td>
<td>813</td>
</tr>
<tr>
<td>exception, chord names</td>
<td>434</td>
</tr>
<tr>
<td>expanding music</td>
<td>54</td>
</tr>
<tr>
<td>explicitClefVisibility</td>
<td>645</td>
</tr>
<tr>
<td>explicitKeySignatureVisibility</td>
<td>645</td>
</tr>
<tr>
<td>Explicitly instantiating voices</td>
<td>175, 177</td>
</tr>
<tr>
<td>expression in MIDI</td>
<td>541</td>
</tr>
<tr>
<td>expression, markup</td>
<td>252</td>
</tr>
<tr>
<td>Expressive marks</td>
<td>59, 125, 131, 133, 136, 138, 139, 140, 145, 148, 151, 248, 398</td>
</tr>
<tr>
<td>\extraChord</td>
<td>286</td>
</tr>
<tr>
<td>extra natural sign, preventing</td>
<td>7</td>
</tr>
<tr>
<td>extra voice for handling breaks</td>
<td>562</td>
</tr>
<tr>
<td>extra-offset</td>
<td>567</td>
</tr>
<tr>
<td>\eyeglasses</td>
<td>759</td>
</tr>
<tr>
<td>Ez_numbers_ engraver</td>
<td>40</td>
</tr>
</tbody>
</table>
## Appendix E: LilyPond Index

### F

<table>
<thead>
<tr>
<th>Entry</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F                           ..................</td>
<td>125</td>
</tr>
<tr>
<td>F clef                          ..................</td>
<td>17, 706</td>
</tr>
<tr>
<td>fall                           ..................</td>
<td>140</td>
</tr>
<tr>
<td>fall                           ..................</td>
<td>140</td>
</tr>
<tr>
<td>false note, percussion           ..................</td>
<td>409</td>
</tr>
<tr>
<td>featherDurations                ..................</td>
<td>98, 813</td>
</tr>
<tr>
<td>feathered beam                  ..................</td>
<td>98</td>
</tr>
<tr>
<td>fermata                        ..................</td>
<td>114, 122, 769</td>
</tr>
<tr>
<td>fermata, Henze                  ..................</td>
<td>122, 769</td>
</tr>
<tr>
<td>fermata, long                   ..................</td>
<td>122, 769</td>
</tr>
<tr>
<td>fermata, on bar line            ..................</td>
<td>248</td>
</tr>
<tr>
<td>fermata, on multi-measure rest    ..................</td>
<td>63</td>
</tr>
<tr>
<td>fermata, short                  ..................</td>
<td>122, 769</td>
</tr>
<tr>
<td>Ferneyhough hairpin             ..................</td>
<td>129</td>
</tr>
<tr>
<td>Feta font                       ..................</td>
<td>683</td>
</tr>
<tr>
<td>\fff                           ..................</td>
<td>125</td>
</tr>
<tr>
<td>\fff                            ..................</td>
<td>125</td>
</tr>
<tr>
<td>\fff                            ..................</td>
<td>125</td>
</tr>
<tr>
<td>fifth                          ..................</td>
<td>5</td>
</tr>
<tr>
<td>figured bass                    ..................</td>
<td>438</td>
</tr>
<tr>
<td>figured bass, alignment        ..................</td>
<td>437</td>
</tr>
<tr>
<td>figured bass, alteration, position ..................</td>
<td>440</td>
</tr>
<tr>
<td>figured bass, extender lines    ..................</td>
<td>440</td>
</tr>
<tr>
<td>FiguredBass                     ..................</td>
<td>210, 441, 442</td>
</tr>
<tr>
<td>figuredBassPlusDirection        ..................</td>
<td>440</td>
</tr>
<tr>
<td>\figuredmode                    ..................</td>
<td>438, 633</td>
</tr>
<tr>
<td>\figures                        ..................</td>
<td>438, 633</td>
</tr>
<tr>
<td>\fill-line                      ..................</td>
<td>260, 723</td>
</tr>
<tr>
<td>\fill-with-pattern             ..................</td>
<td>516, 723</td>
</tr>
<tr>
<td>\filled-box                    ..................</td>
<td>263, 740</td>
</tr>
<tr>
<td>finals                         ..................</td>
<td>459</td>
</tr>
<tr>
<td>\finals                        ..................</td>
<td>460</td>
</tr>
<tr>
<td>finding available fonts         ..................</td>
<td>271</td>
</tr>
<tr>
<td>\finger                        ..................</td>
<td>229, 712, 813</td>
</tr>
<tr>
<td>finger change                   ..................</td>
<td>229</td>
</tr>
<tr>
<td>finger-interface               ..................</td>
<td>617</td>
</tr>
<tr>
<td>Fingering                      ..................</td>
<td>231, 355, 616, 617</td>
</tr>
<tr>
<td>fingering                      ..................</td>
<td>229</td>
</tr>
<tr>
<td>fingering chart                ..................</td>
<td>412</td>
</tr>
<tr>
<td>fingering chord                ..................</td>
<td>229</td>
</tr>
<tr>
<td>fingering instruction, for chords ..................</td>
<td>229</td>
</tr>
<tr>
<td>fingering vs. string number     ..................</td>
<td>354</td>
</tr>
<tr>
<td>fingering, adding to fret diagram ..................</td>
<td>395</td>
</tr>
<tr>
<td>fingering, and multi-measure rest ..................</td>
<td>66</td>
</tr>
<tr>
<td>fingering, inside of staff      ..................</td>
<td>230</td>
</tr>
<tr>
<td>fingering, orientation        ..................</td>
<td>230</td>
</tr>
<tr>
<td>fingering, right hand, for fretted instrument ..................</td>
<td>396</td>
</tr>
<tr>
<td>fingering, right-hand, placement ..................</td>
<td>397</td>
</tr>
<tr>
<td>fingering, stem support        ..................</td>
<td>230</td>
</tr>
<tr>
<td>fingering, symbols for wind instruments ..................</td>
<td>411</td>
</tr>
<tr>
<td>fingering, thumb               ..................</td>
<td>122, 229, 769</td>
</tr>
<tr>
<td>fingering-event                ..................</td>
<td>231, 617</td>
</tr>
<tr>
<td>Fingering engraver             ..................</td>
<td>231, 616, 617, 619</td>
</tr>
<tr>
<td>FingeringEvent                 ..................</td>
<td>231, 617</td>
</tr>
<tr>
<td>fingeringOrientations          ..................</td>
<td>230</td>
</tr>
<tr>
<td>first-page-number              ..................</td>
<td>555</td>
</tr>
<tr>
<td>\first-visible                 ..................</td>
<td>759</td>
</tr>
<tr>
<td>\fixed                         ..................</td>
<td>2, 813</td>
</tr>
<tr>
<td>Fixing overlapping notation    ..................</td>
<td>341, 342</td>
</tr>
</tbody>
</table>

### F  flag ............................................. 488, 454

### F  flag, mensural  .................................. 454

### F  \\flageolet ........................................... 122, 411, 769

### F  \flageolet, changing size  ...................... 411

### F  flags, flat  ........................................ 97

### F  flared hairpin  ................................... 129

### F  flat  ............................................. 7

### F  flat  ............................................. 6

### F  \flat  ............................................ 745

### F  flat flags  ....................................... 97

### F  flat, double  ..................................... 6

### F  Flex ............................................. 774

### F  \flexa ........................................... 468

### F  Flexible architecture  .......................... 244, 245

### F  floor tom  ....................................... 770

### F  follow voice  ..................................... 344

### F  followVoice  ...................................... 344

### F  font  ............................................ 773

### F  font family ...................................... 256

### F  font family, setting  .......................... 271

### F  font size ........................................ 254

### F  font size (notation)  .......................... 225

### F  font size (notation), scaling .................. 225

### F  font size (notation), standard ................. 228

### F  font size, setting  ............................. 559

### F  font switching  .................................. 254

### F  font, changing for entire document .......... 271

### F  font, Emmentaler  ............................... 683

### F  font, explained  .................................. 267

### F  font, external files  ............................ 267

### F  font, Feta ........................................ 683

### F  font, finding available ......................... 271

### F  font, music notation  ........................... 268

### F  font, non-text in markup ......................... 267

### F  font, Parmesan  ................................... 683

### F  font-encoding  .................................... 268

### F  font-interface  ................................. 228, 617, 761

### F  font-interface  .................................... 228

### F  font-size ......................................... 225, 228

### F  \fontCaps ......................................... 712

### F  \fontsize .......................................... 254, 712

### F  font-size ......................................... 225

### F  foot mark  ....................................... 122, 769

### F  footer ............................................ 492

### F  footer, page  ...................................... 498

### F  footnotenote  ..................................... 505

### F  \footnote ............................................ 505, 759, 813

### F  footnote, event-based  .......................... 506

### F  footnote, in music expressions ................. 505

### F  footnote, in stand-alone text .................. 510

### F  footnote, time-based  ............................ 507

### F  footnote-separator-markup  ...................... 556

### F  Footnote engraver  ................................ 513

### F  FootnoteEvent  ..................................... 513

### F  FootnoteItem  ...................................... 513

### F  FootnoteSpanner  .................................. 513

### F  Forbid_line_break engraver ...................... 81

### F  Forbid_line_break engraver ...................... 53

### F  forget ............................................. 35

### F  forget, accidental style  ...................... 35

### F  format, rehearsal mark  ......................... 114

### F  Formatting text  .................................. 799, 808

### F  formatting, cue notes  ........................... 217

### F  formatting, in lyrics  ........................... 275
formatting, text spanner ........................................ 247
four-bar music .................................................. 562
Four-part SATB vocal score .................................... 315
four-string-banjo ............................................... 401
\frac ......................................................... 125
\fraction ....................................................... 759
fragment ......................................................... 217
fragment, music ................................................ 527
fragment, quoting ............................................... 214
framing text .................................................... 262
\freeBass ...................................................... 754
French clef ..................................................... 17, 706
\frenchChords ................................................ 432
Frenched score ................................................ 318
Frenched score ................................................ 205
\frenched staff ............................................. 205, 210
\frenched staff ............................................. 202, 205
Frenched staves ................................................ 318
fret ............................................................. 357
fret diagram .................................................... 373, 383
fret diagram, adding custom .................................. 386
fret diagram, adding fingering ............................... 395
fret diagram, automatic ....................................... 393
fret diagram, changing orientation ......................... 381
fret diagram, custom ......................................... 374
fret diagram, customized ..................................... 381, 382
fret diagram, left-handed .................................... 376
fret diagram, mandolin ....................................... 383
fret diagram, markup ......................................... 374
fret diagram, transposing .................................... 385
fret diagram, ukulele ......................................... 383
fret diagram, with chord names ............................. 385
\fret-diagram ................................................ 374, 750
\fret-diagram-interface .................................... 381, 383, 388, 393, 396
\fret-diagram-interface ................................... 381
\fret-diagram-terse ......................................... 377, 750
fret-diagram-terse markup .................................. 377
\fret-diagram-verbos ....................................... 378, 751
fret-diagram-verbos markup ................................ 378
fretboard fret diagram, customized ......................... 388
fretboard, alternate table .................................... 392
fretboard, define predefined ................................ 389
FretBoards ..................................................... 383
fretted instrument, chord shape ......................... 387
fretted instrument, dampened note ......................... 398
fretted instrument, harmonics .............................. 398
fretted instrument, indicating position and barring .... 398
fretted instrument, predefined string tunings ............ 370
fretted instrument, right-hand fingering ................ 396
Fretted strings ........................................... 355, 369, 373, 383, 393, 396, 398, 400, 401
\fromproperty ................................................. 759
full score ...................................................... 317, 318
full-measure rest ............................................. 62
funk shape note head ........................................ 42
\funkHeads ................................................... 42
\funkHeadsMinor ............................................. 43

G

G clef .......................................................... 17
\general-align ............................................... 258, 724
\germanChords .............................................. 432
ghost note ...................................................... 234
ghost note, percussion ....................................... 409
glissando ........................................................ 145
Glissando ...................................................... 145, 641
glissando ....................................................... 140
\glissando ..................................................... 140
glissando, across repeats .................................... 143
glissando, and repeats ....................................... 158
glissando, breakable ......................................... 143
glissando, chords in tablatures ............................ 367
glissando, contemporary ..................................... 142
glissando, timing marks ..................................... 142
\glissandoMap ................................................ 141
global variable ................................................ 775
glyph ............................................................ 773
glyph, music .................................................... 114
\grace ......................................................... 115, 813
grace note ...................................................... 115, 413
grace note, and lyrics ........................................ 309
grace note, and strict spacing ............................... 118
grace note, changing layout settings ..................... 117
grace note, following ......................................... 116
grace note, synchronization .................................. 119
grace note, tweaking ......................................... 117
\graceNotes .................................................... 119
Grace_auto_beam_engraver .................................. 119
Grace_beam_engraver ......................................... 119
Grace_ engraver ................................................ 119
\grace_spacing_ engraver ................................... 119
GraceMusic ..................................................... 119
grammar, for LilyPond ....................................... 774
grand staff ..................................................... 196
grand staff ..................................................... 193
GrandStaff .................................................... 35, 196
graphic notation .............................................. 263
graphic object, drawing ...................................... 262
graphic object, embedding ................................... 262
graphical object ............................................... 773
graphical object interface .................................... 774
Graphical Object Interfaces ................................. 774
graphics, embedding ......................................... 262, 263
Gregorian accidental ......................................... 459
Gregorian articulation ......................................... 460
gregorian articulation, accentus ......................... 460, 770
gregorian articulation, circle ................................ 460, 770
gregorian articulation, epismFinis ......................... 460
gregorian articulation, epismInitium ..................... 460
gregorian articulation, ictus ............................... 460, 770
gregorian articulation, semicircle ......................... 460, 770
Gregorian key signature ...................................... 459
Gregorian music, modern transcription .................. 332
Gregorian square neumes ligature ......................... 461
Gregorian transcription staff ................................ 191
GregorianTranscriptionStaff ................................ 192
GregorianTranscriptionStaff ................................ 191
grid line ........................................................ 238
grid lines, changing appearance ........................... 239
grid-line-interface ........................................... 240
grid-point-interface .......................................... 240
Grid_line_span_engraver ..................................... 240
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ledger line</td>
<td>202</td>
</tr>
<tr>
<td>ledger line, internal</td>
<td>199</td>
</tr>
<tr>
<td>ledger line, modifying</td>
<td>199</td>
</tr>
<tr>
<td>ledger-line-spanner-interface</td>
<td>40</td>
</tr>
<tr>
<td>Ledger_line_engraver</td>
<td>40</td>
</tr>
<tr>
<td>LedgerLineSpanner</td>
<td>40</td>
</tr>
<tr>
<td>\left-align</td>
<td>257, 729</td>
</tr>
<tr>
<td>left-aligning text</td>
<td>729</td>
</tr>
<tr>
<td>\left-brace</td>
<td>760</td>
</tr>
<tr>
<td>\left-column</td>
<td>729</td>
</tr>
<tr>
<td>left-handed fret diagram</td>
<td>576</td>
</tr>
<tr>
<td>left-margin</td>
<td>552</td>
</tr>
<tr>
<td>Length and thickness of objects</td>
<td>205, 636</td>
</tr>
<tr>
<td>length of multi-measure rest</td>
<td>64</td>
</tr>
<tr>
<td>length of note</td>
<td>47</td>
</tr>
<tr>
<td>lexers</td>
<td>774</td>
</tr>
<tr>
<td>line-way</td>
<td>122, 769</td>
</tr>
<tr>
<td>ligature</td>
<td>447, 449, 457, 468, 471</td>
</tr>
<tr>
<td>ligature, in text</td>
<td>722</td>
</tr>
<tr>
<td>ligature, Kievan</td>
<td>471</td>
</tr>
<tr>
<td>ligature, mensural</td>
<td>456</td>
</tr>
<tr>
<td>ligature, square neumes</td>
<td>461</td>
</tr>
<tr>
<td>LilyPond grammar</td>
<td>775</td>
</tr>
<tr>
<td>LilyPond grammar</td>
<td>774</td>
</tr>
<tr>
<td>line</td>
<td>202</td>
</tr>
<tr>
<td>line break</td>
<td>100</td>
</tr>
<tr>
<td>line break, beams</td>
<td>85</td>
</tr>
<tr>
<td>line break, cadenzas</td>
<td>77</td>
</tr>
<tr>
<td>line break, managing with extra voice</td>
<td>562</td>
</tr>
<tr>
<td>line break, unmeasured music</td>
<td>77</td>
</tr>
<tr>
<td>line comment</td>
<td>487, 491</td>
</tr>
<tr>
<td>line, cross-staff</td>
<td>344</td>
</tr>
<tr>
<td>line, end of, positioning marks</td>
<td>249</td>
</tr>
<tr>
<td>line, grid</td>
<td>238</td>
</tr>
<tr>
<td>line, staff-change</td>
<td>344</td>
</tr>
<tr>
<td>line, staff-change follower</td>
<td>344</td>
</tr>
<tr>
<td>line, to indicate duration</td>
<td>444</td>
</tr>
<tr>
<td>line-spanner-interface</td>
<td>641</td>
</tr>
<tr>
<td>line-width</td>
<td>260, 551, 587</td>
</tr>
<tr>
<td>\lines</td>
<td>461, 468</td>
</tr>
<tr>
<td>LineBreakEvent</td>
<td>563</td>
</tr>
<tr>
<td>\linbreak</td>
<td>122, 769</td>
</tr>
<tr>
<td>lines, vertical, between staves</td>
<td>238</td>
</tr>
<tr>
<td>list of colors</td>
<td>681</td>
</tr>
<tr>
<td>list of keys in woodwind diagrams</td>
<td>422</td>
</tr>
<tr>
<td>list of woodwind diagrams</td>
<td>418</td>
</tr>
<tr>
<td>listing available fonts</td>
<td>271</td>
</tr>
<tr>
<td>Literature list</td>
<td>432, 436</td>
</tr>
<tr>
<td>locrian</td>
<td>22</td>
</tr>
<tr>
<td>\locrian</td>
<td>22</td>
</tr>
<tr>
<td>long fermata</td>
<td>122, 769</td>
</tr>
<tr>
<td>longa</td>
<td>49, 61</td>
</tr>
<tr>
<td>longa rest</td>
<td>59</td>
</tr>
<tr>
<td>longfermata</td>
<td>122, 769</td>
</tr>
<tr>
<td>lookup</td>
<td>760</td>
</tr>
<tr>
<td>low bongo</td>
<td>770</td>
</tr>
<tr>
<td>low conga</td>
<td>770</td>
</tr>
<tr>
<td>low timbale</td>
<td>770</td>
</tr>
<tr>
<td>low tom tom</td>
<td>770</td>
</tr>
<tr>
<td>\lower</td>
<td>258, 730</td>
</tr>
<tr>
<td>'lower mordent' ornament</td>
<td>122, 768</td>
</tr>
<tr>
<td>lowering text</td>
<td>730</td>
</tr>
<tr>
<td>\lto</td>
<td>122, 769</td>
</tr>
<tr>
<td>\lubtabulate</td>
<td>401</td>
</tr>
<tr>
<td>lute tuning</td>
<td>402</td>
</tr>
<tr>
<td>\ly::add-context-mod</td>
<td>824</td>
</tr>
<tr>
<td>\ly::add-file-name-alist</td>
<td>824</td>
</tr>
<tr>
<td>\ly::add-interface</td>
<td>825</td>
</tr>
<tr>
<td>\ly::add-listener</td>
<td>825</td>
</tr>
<tr>
<td>\ly::add-option</td>
<td>825</td>
</tr>
<tr>
<td>\ly::all-grob-interfaces</td>
<td>825</td>
</tr>
<tr>
<td>\ly::all-options</td>
<td>825</td>
</tr>
<tr>
<td>\ly::all-stencil-expressions</td>
<td>825</td>
</tr>
<tr>
<td>\ly::angle</td>
<td>825</td>
</tr>
<tr>
<td>\ly::assoc-get</td>
<td>825</td>
</tr>
<tr>
<td>\ly::axis-group-interface:add-element</td>
<td>825</td>
</tr>
<tr>
<td>\ly::basic-progress</td>
<td>825</td>
</tr>
<tr>
<td>\ly::beam-score-count</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-add-bookpart!</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-add-score!</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-book-parts</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-header</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-paper</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-process</td>
<td>825</td>
</tr>
<tr>
<td>\ly::book-process-to-systems</td>
<td>826</td>
</tr>
<tr>
<td>\ly::book-scores</td>
<td>826</td>
</tr>
<tr>
<td>\ly::book-set-header</td>
<td>826</td>
</tr>
<tr>
<td>\ly::book?</td>
<td>825</td>
</tr>
<tr>
<td>\ly::box?</td>
<td>826</td>
</tr>
<tr>
<td>\ly::bp</td>
<td>826</td>
</tr>
<tr>
<td>\ly::bracket</td>
<td>826</td>
</tr>
<tr>
<td>\ly::broadcast</td>
<td>826</td>
</tr>
<tr>
<td>\ly::camel-case-&gt;lisp-identifier</td>
<td>826</td>
</tr>
<tr>
<td>\ly::chain-assoc-get</td>
<td>826</td>
</tr>
<tr>
<td>\ly::check-expected-warnings</td>
<td>826</td>
</tr>
<tr>
<td>\ly::cm</td>
<td>826</td>
</tr>
<tr>
<td>\ly::command-line-code</td>
<td>826</td>
</tr>
<tr>
<td>\ly::command-line-options</td>
<td>826</td>
</tr>
<tr>
<td>\ly::connect-dispatchers</td>
<td>826</td>
</tr>
<tr>
<td>\ly::context-current-moment</td>
<td>826</td>
</tr>
<tr>
<td>\ly::context-def-lookup</td>
<td>827</td>
</tr>
<tr>
<td>\ly::context-def-modify</td>
<td>827</td>
</tr>
</tbody>
</table>
note continuation ........................................... 444
note duration ............................................. 47
note duration, default ...................................... 47
note grouping bracket ....................................... 240
note head ................................................. 453, 470
note head .................................................. 225
note head style ............................................. 39, 705
note head, Aiken .......................................... 42
note head, Aiken, thin variant ............................. 44
note head, ancient ........................................ 453, 469
note head, Christian Harmony ............................. 42
note head, cross ............................................ 39
note head, diamond ......................................... 39
note head, diamond-shaped ................................ 352
note head, easy notation .................................. 40
note head, easy play ....................................... 40
note head, funk ............................................. 42
note head, guitar .......................................... 39
note head, Harmonia Sacra ................................ 42
note head, harmonic ........................................ 39
note head, improvisation ................................... 45
note head, parlando ......................................... 39
note head, practice ......................................... 40
note head, Sacred Harp ..................................... 42
note head, shape ............................................. 42
note head, slashed .......................................... 45
note head, Southern Harmony ................................ 42
note head, special ......................................... 39
note head, Walker .......................................... 42
note length .................................................. 47
note name, Arabic ......................................... 479
note name, default ......................................... 6
note name, Dutch .......................................... 6
note name, Hel-arabic ...................................... 479
note name, other languages ................................ 8
note name, printing ........................................ 236
note pitch, default ......................................... 47
note value .................................................... 52
note, colored ............................................... 232
note, colored in chords .................................... 233
note, cross-staff ........................................... 340, 345
note, dotted ............................................... 48
note, dotted, moving horizontally ......................... 181
note, double-dotted ......................................... 48
note, ghost .................................................. 234
note, hidden ................................................ 231
note, horizontal shift ...................................... 182
note, invisible .............................................. 231
note, parenthesized ........................................ 234
note, smaller ............................................... 217
note, spacing horizontally ................................ 584
note, splitting .............................................. 80
note, transparent .......................................... 231
note, transpose ............................................. 231
note, within text, by duration ............................ 746
note, within text, by log and dot-count .................. 746
\note-by-number ........................................... 746
note-collision-interface .................................. 795, 798, 800
note-event .................................................. 40, 42, 45
\note-event ................................................ 216
note-head-interface ....................................... 40, 42, 45
NoteHead_line_engraver .................................... 345
NoteHeads_engraver ....................................... 40, 42, 45, 81, 614
Note_name_engraver ....................................... 80
Note_name_engraver ....................................... 237
Note_name_engraver ....................................... 237
NoteSpacing .............................................. 232, 583
\null ....................................................... 258, 761
NullVoice .................................................. 304
\number ..................................................... 75
\number, bar ............................................... 107
\number, measure ......................................... 107
\numericTimeSignature ................................... 67

O
object, colored ............................................. 232
object, coloring .......................................... 643
object, markup ............................................ 243
object, overwriting ........................................ 643
object, rotating ........................................... 649
object, visibility .......................................... 641

Objects and interfaces .................................. 513, 773, 774
octavation ................................................ 27
octavation ................................................ 24
octave changing mark ..................................... 1
octave check ............................................... 10
octave correction ......................................... 10
octave entry, absolute ................................... 715
octave entry, relative .................................... 2
octave specification, absolute ........................... 1
octave specification, relative ............................ 2
octave transposition ...................................... 17
octave transposition, optional ......................... 17
\octaveCheck ............................................. 10, 815
\offset .................................................... 626, 815
offseting .................................................. 626
\omit ....................................................... 642, 815
On the un-nestedness of brackets and ties ............ 136, 138
\on-the-fly ................................................ 503, 761
\once ....................................................... 621, 623, 628, 660, 815
\oneVoice ................................................ 173
\open ....................................................... 351, 411, 769
\open' articulation ...................................... 122, 351, 411, 769
open bongo ............................................... 770
open conga ............................................... 770
open high hat ............................................. 122, 770
open string indication .................................... 351
open timbre .............................................. 770
opera ....................................................... 317
operation, inversion ...................................... 14
operation, modal .......................................... 15
operation, modal inversion ............................... 16
operation, retrograde .................................... 14
operation, transposition ................................ 15

Optical spacing ........................................... 583
optional octave transposition ............................ 17
oratorio .................................................... 314
Appendix E: LilyPond index

overrideProperty ........................................ 619
\overrideProperty ....................................... 626, 816
\overrideTimeSignatureSettings ....................... 67, 816
overriding for only one moment ....................... 623
overriding property within text markup ............... 761
\overtie .................................................. 715
overtie-ing text .......................................... 715
overwriting object ....................................... 643

p
\p ........................................................................ 125
\pad-around .................................................. 262, 730
\pad-markup ................................................... 262, 730
\pad-to-box ................................................... 262, 731
\pad-x .......................................................... 262, 731
padding ......................................................... 618
padding around text ......................................... 262
padding in markup, horizontal ......................... 258
padding in markup, vertical ............................... 259
padding text ................................................... 250
padding text horizontally .................................. 731
page break ..................................................... 587
page break, cadenzas ....................................... 77
page break, managing with extra voice ............... 562
page break, unmetered music ............................ 77
page breaking, manual .................................... 563
page footer ..................................................... 498
page header ................................................... 498
page layout ..................................................... 587
page number, auto-numbering ......................... 555
page number, in roman numerals ..................... 555
page number, specify first ............................... 555
page number, suppress ................................... 555
page numbers, referencing .............................. 513
page size ....................................................... 547
page, orientation ............................................ 548
page-breaking ............................................... 554
page-breaking-system-system-spacing ............... 554
\page-count ................................................... 554
\page-link ..................................................... 761
\page-number-type ......................................... 555
\page-ref ...................................................... 514, 762
\page-spacing-weight ..................................... 555
\pageBreak .................................................... 563, 816
\pageTurn ...................................................... 566, 816
\palmMute ..................................................... 816
\palmMuteOn ............................................... 816
\pan position in MIDI ..................................... 541
Pango .......................................................... 267
\paper ........................................................ 490, 547
paper size .................................................... 547
paper size, landscape ..................................... 548
paper size, orientation ................................... 548
paper-height .................................................. 549
paper-width .................................................. 551
parallel music ................................................. 188
\parallelMusic ............................................... 188, 816
parentheses-interface ..................................... 234
ParenthesesItem ............................................. 234
parenthesis ................................................... 234
Parenthesis_ engraver .................................... 234
\parenthesize ................................................ 204, 741, 816
parenthesized accidental ................................. 6
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parlando</td>
<td>324</td>
</tr>
<tr>
<td>parlando note head</td>
<td>39</td>
</tr>
<tr>
<td>Parmesan font</td>
<td>683</td>
</tr>
<tr>
<td>parser</td>
<td>774</td>
</tr>
<tr>
<td>parser variable</td>
<td>775</td>
</tr>
<tr>
<td><em>part</em></td>
<td>188</td>
</tr>
<tr>
<td>part combiner</td>
<td>183</td>
</tr>
<tr>
<td>part combiner, changing text</td>
<td>187</td>
</tr>
<tr>
<td>part song</td>
<td>314</td>
</tr>
<tr>
<td><code>\partCombine</code></td>
<td>183, 304, 816</td>
</tr>
<tr>
<td><code>\partCombine and lyrics</code></td>
<td>186, 304</td>
</tr>
<tr>
<td><code>\partCombineApart</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineAutomatic</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineChords</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineDown</code></td>
<td>817</td>
</tr>
<tr>
<td><code>\partCombineForce</code></td>
<td>817</td>
</tr>
<tr>
<td><code>\partCombineListener</code></td>
<td>775</td>
</tr>
<tr>
<td><code>PartCombineMusic</code></td>
<td>188</td>
</tr>
<tr>
<td><code>\partCombineSoloI</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineSoloII</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineUnisono</code></td>
<td>185</td>
</tr>
<tr>
<td><code>\partCombineUp</code></td>
<td>817</td>
</tr>
<tr>
<td><code>\partial</code></td>
<td>74, 151, 153, 817</td>
</tr>
<tr>
<td>partial measure</td>
<td>74</td>
</tr>
<tr>
<td>partial measure, in hymns</td>
<td>336</td>
</tr>
<tr>
<td>parts, desk</td>
<td>210</td>
</tr>
<tr>
<td>parts, individual</td>
<td>210</td>
</tr>
<tr>
<td>parts, section</td>
<td>210</td>
</tr>
<tr>
<td><code>\path</code></td>
<td>741</td>
</tr>
<tr>
<td>path, drawing</td>
<td>741</td>
</tr>
<tr>
<td><code>\pattern</code></td>
<td>762</td>
</tr>
<tr>
<td>pause mark</td>
<td>138</td>
</tr>
<tr>
<td>PDF metadata</td>
<td>504</td>
</tr>
<tr>
<td>pedal diagram, harp</td>
<td>349</td>
</tr>
<tr>
<td>pedal high hat</td>
<td>122, 407, 770</td>
</tr>
<tr>
<td>pedal indication style</td>
<td>347</td>
</tr>
<tr>
<td>pedal indication, bracket</td>
<td>347</td>
</tr>
<tr>
<td>pedal indication, mixed</td>
<td>347</td>
</tr>
<tr>
<td>pedal indication, text</td>
<td>347</td>
</tr>
<tr>
<td>pedal mark, heel</td>
<td>122, 769</td>
</tr>
<tr>
<td>pedal mark, organ</td>
<td>122, 769</td>
</tr>
<tr>
<td>pedal mark, toe</td>
<td>122, 769</td>
</tr>
<tr>
<td>pedal sustain style</td>
<td>347</td>
</tr>
<tr>
<td>pedal, harp</td>
<td>349</td>
</tr>
<tr>
<td>pedal, piano</td>
<td>347</td>
</tr>
<tr>
<td>pedal, sostenuto</td>
<td>347</td>
</tr>
<tr>
<td>pedal, sustain</td>
<td>347</td>
</tr>
<tr>
<td><code>pedalSustainStyle</code></td>
<td>347</td>
</tr>
<tr>
<td><code>percent</code></td>
<td>163</td>
</tr>
<tr>
<td><code>percent repeat</code></td>
<td>165</td>
</tr>
<tr>
<td><code>percent repeat</code></td>
<td>165</td>
</tr>
<tr>
<td><code>percent repeat, count visibility</code></td>
<td>164</td>
</tr>
<tr>
<td><code>percent repeat, isolated</code></td>
<td>164</td>
</tr>
<tr>
<td><code>Percent_repeat_engraver</code></td>
<td>165</td>
</tr>
<tr>
<td><code>PercentRepeat</code></td>
<td>165</td>
</tr>
<tr>
<td><code>PercentRepeatCounter</code></td>
<td>165</td>
</tr>
<tr>
<td><code>PercentRepeatedMusic</code></td>
<td>165</td>
</tr>
<tr>
<td><code>Percussion</code></td>
<td>403, 404, 407, 409</td>
</tr>
<tr>
<td>percussion</td>
<td>403, 404</td>
</tr>
<tr>
<td>percussion clef</td>
<td>403, 706</td>
</tr>
<tr>
<td>percussion staff</td>
<td>191</td>
</tr>
<tr>
<td>percussion, custom</td>
<td>407</td>
</tr>
<tr>
<td>percussion, dead note</td>
<td>409</td>
</tr>
<tr>
<td>percussion, false note</td>
<td>409</td>
</tr>
</tbody>
</table>

- Percussion, ghost note: 409
- Percussion, muted note: 409
- Percussion, silenced note: 409
- Persian makam: 478
- Petrucci: 447, 448
- Petrucci clef: 451, 706
- Phrasing bracket: 240
- Phrasing mark: 136
- Phrasing slur: 134, 136
- Phrasing slur, dashed: 137
- Phrasing slur, defining dash patterns: 137
- Phrasing slur, dotted: 137
- Phrasing slur, half solid and half dashed: 137
- Phrasing slur, multiple: 136
- Phrasing slur, simultaneous: 136
- Phrasing, in lyrics: 282
- PhrasingSlur: 138
- `\phrasingSlurDashPattern`: 137, 817
- `\phrasingSlurDotted`: 137
- `\phrasingSlurDown`: 136
- `\phrasingSlurHalfDashed`: 137
- `\phrasingSlurHalfSolid`: 137
- `\phrasingSlurNeutral`: 136
- `\phrasingSlurSolid`: 137
- `\phrasingSlurUp`: 136
- `\phrygian`: 22
- Piano: 32
- Piano accidental style: 32
- Piano music, centering dynamics: 339
- Piano pedal: 347
- Piano staff: 193, 339
- Piano-cautionary: 32
- Piano-cautionary accidental style: 32
- Piano_pedal_engraver: 347
- PianoPedalBracket: 347
- PianoStaff: 35, 148, 196, 213, 315, 382
- PianoStaff: 339, 342
- Pickup measure: 74
- Pick up, in a repeat: 153
- Pitch: 1
- Pitch name: 1
- Pitch name, other languages: 8
- Pitch names: 2, 5, 7, 10, 455
- Pitch range: 36
- Pitch, default: 47
- Pitch, isolated: 47
- Pitch, transposition of: 11
- Pitch_squash_engraver: 46, 84, 614, 786
- Pitched trill: 149
- Pitched trill, with accidental: 150
- `\pitchedTrill`: 149, 817
- Pitches: 2, 5, 7, 10, 11, 14, 21, 24, 27, 28, 35, 38, 40, 42, 45, 46, 452, 482
- Pitches and key signatures: 6, 7, 22, 24, 482
- Pitches, ’smart’ transposition: 12
- Pitchnames: 775
- Pizzicato, Bartók: 353
- Pizzicato, snap: 353
- Placeholder event: 169
- Placement of objects: 123, 125, 246
- Placement, lyrics: 289
placement, right-hand fingering ........................................ 397
placing horizontal brackets, around text ......................... 740
placing parentheses, around text .................................... 741
placing vertical brackets, around text ........................... 736
\pointAndClickOff .................................................. 817
\pointAndClickOn .................................................. 817
\pointAndClickTypes ............................................. 817
\polymetric .......................................................... 54, 80
polymetric meter, with beams ...................................... 77
polymetric score ...................................................... 606
polymetric signatures ................................................ 77
polymetric time signature .......................................... 80
polyphonic music .................................................... 178
polyphony .............................................................. 182
polyphony, additional voices ....................................... 181
polyphony, in tablatures ............................................. 362
polyphony, shared lyrics ........................................... 304
polyphony, single-staff ............................................. 173
\portato .............................................................. 125
\portato .............................................................. 123
\quaverarticulation .................................................. 122, 768
position, figured bass alteration ................................. 440
position, lyrics ....................................................... 204
position, multi-measure rest ....................................... 64
position, ossia ......................................................... 204
post-events ........................................................... 768
\postscrip ............................................................ 263
\postscript ............................................................ 263, 742
\powerchord ........................................................ 401
\powerchord ........................................................ 400
\pp ................................................................. 125
\ppp ................................................................. 125
\pppp ................................................................. 125
\ppppp ................................................................. 125
\practicehead ......................................................... 40
\prall ................................................................. 122, 768
\pralldown .......................................................... 122, 768
\prallmordent ....................................................... 122, 768
\prallprall .......................................................... 122, 768
\pralltriller' ornament ............................................ 122, 768
\pralltriller, down' ornament .................................... 122, 768
\pralltriller, long' ornament ..................................... 122, 768
\pralltriller, up' ornament ....................................... 122, 768
\prallup .............................................................. 122, 768
predefined string tuning, for
fretted instruments ................................................... 370
\predefinedDiagramTable ......................................... 392
\predefinedFretboardsOff ......................................... 394
\predefinedFretboardsOn .......................................... 394
prima volta ........................................................... 151
\print-all-headers .................................................. 556
\print-first-page-number ......................................... 555
\print-page-number ................................................ 555
\printAccidentalNames ............................................. 236
\printChordName .................................................... 429
\printOrder .......................................................... 643
\printing .............................................................. 253
\printSpecialChar ................................................... 253
\printNotesLanguage ................................................. 236
\printOctaveNames .................................................. 236
\prob ................................................................. 775
\Propertiesfoundininterfaces ....................................... 774
\Propertiesoflayoutobjects ....................................... 773
\property ............................................................. 620
\propertyobject ..................................................... 775
\propertygrobs ....................................................... 621
\property-recursive ................................................ 762
\property-override .................................................. 817
\propertyRevert ....................................................... 817
\PropertySet .......................................................... 619
\propertySet .......................................................... 817
\propertyTweak ....................................................... 817
\propertyUnset ....................................................... 818
psalm ................................................................. 326, 331
\pt ................................................................. 635
pull off ............................................................... 368
punctuation, in lyrics .............................................. 275
purify container, Scheme ........................................... 661
\pushToTag ............................................................ 523, 818
\put-adjacent ........................................................ 731
putting space around text ......................................... 730
\Q ................................................................. 214, 217, 818
quote, in lyrics ....................................................... 275, 282
\quote ................................................................. 214
\quoted ............................................................... 245
\quotedtext .......................................................... 253
\quotedtext, in markup mode ..................................... 216
\quotedEventTypes .................................................. 216
\quoteDuring ........................................................ 818
\QuoteMusic ............................................................ 217
\R ................................................................. 214, 217, 818
\Realmusicexample .................................................. 180, 182, 339
\reduceChords .......................................................... 82, 818
reversing context .................................................... 599
\referencing .......................................................... 765
\referencingpage ..................................................... 348
\referred .............................................................. 562
\rail ................................................................. 113
\raster ................................................................. 482
\Ratsibona, Editio ................................................... 447
\Realmusicexample .................................................. 180, 182, 339
\reduceChords .......................................................... 82, 818
\referencingcontext .................................................. 599
\referencingpagelabel, in text .................................... 765
\referencingpagenumber, in text ................................ 761, 762
\register ............................................................... 348
\regular ............................................................... 562
\rehearsal ............................................................. 113
\rehearsalmark ......................................................... 73
\rehearsalmarkbelow ................................................. 33
\rehearsalmarkformat ............................................... 114
\rehearsalmarkmanual ............................................... 114
\rehearsalmarkstyle .................................................. 114
\rehearsalmark ......................................................... 119, 251
Appendix E: LilyPond index

relative ........................................... 2
\relative ...................................... 2, 5, 14, 343, 818
relative music, and \autoChange .............. 343
relative octave entry .......................... 2
relative octave entry, and chords ............ 4
relative octave entry, and transposition ... 5
relative octave specification ................. 2
relative pitch, chords ......................... 169
RelativeOctaveCheck .......................... 11
RelativeOctaveMusic ......................... 5
religious music .................................. 326
reminder accidental ............................ 6
removal, in chord ............................... 427
\remove ........................................ 605
remove tagged music ............................ 520
\remove-empty .................................. 206
\remove-first .................................... 206
\remove-grace-property ....................... 117
\remove-layer ................................... 208
\RemoveAllEmptyStaves ....................... 205, 821
\RemoveEmptyStaves ......................... 205, 822
\removeWithTag ................................ 520, 818
removing bar numbers ......................... 111
removing cue notes ............................ 221
renaissance music ............................ 195
\repeat ........................................... 158
\repeat .......................................... 102
\repeat ........................................... 151
repeat bar ....................................... 99
repeat number, changing ...................... 159
\repeat \percent ................................ 163
\repeat tremolo ................................ 166
repeat volta, changing ....................... 159
repeat, alternative ............................ 161
repeat, alternative bar numbers .......... 157
repeat, ambiguous ............................. 158
repeat, and glissandi ......................... 143, 158
repeat, and lyrics .............................. 295
repeat, and measure number ................. 158
repeat, and slur ................................ 158
repeat, bar numbers with letters .......... 157
repeat, double, style for volta ............. 157
repeat, end ....................................... 159
repeat, manual .................................. 159
repeat, measure ................................ 163
repeat, nested .................................. 158
repeat, normal .................................. 151
repeat, percent ................................ 163
repeat, percent counter ...................... 164
repeat, percent, count visibility .......... 164
repeat, percent, isolated ..................... 164
repeat, short ................................... 163
repeat, start .................................... 159
repeat, timing information .................. 158
repeat, tremolo ................................ 166
repeat, unfold .................................. 161
repeat, with alternate endings .............. 151
repeat, with anacrusis ......................... 153
repeat, with bar checks ....................... 153
repeat, with pickup ............................ 153
repeat, with segno ............................. 154
repeat, with ties ............................... 154
repeat, with upbeat ............................ 153
repeat, written-out ............................ 161
\repeatCommands ............................... 159
\repeatCountVisibility ...................... 164
repeated chords, suppressing ............... 391, 431
RepeatedMusic ................................ 158, 161, 162
repeating lyrics, with alternative endings . 299
repeating tie .................................. 56
Repeats ......................................... 158, 161, 162, 165, 167
repeats in MIDI ............................... 538
RepeatSlash ..................................... 165
\RepeatSlashEvent .................. 165
\repeatTie ...................................... 56, 154, 300
repetition, using q ............................ 170, 357
\replace ....................................... 716
reserved character, printing ................. 253
\resetRelativeOctave ........................ 5, 818
resizing of staves ............................ 202
\responsum ..................................... 462
Rest ............................................. 61
rest .............................................. 59
rest, ancient ................................... 454
rest, church ..................................... 224
rest, collisions of ............................. 66
rest, condensing ordinary .................... 66
rest, entering durations ...................... 59
rest, full-measure ............................. 62
rest, invisible .................................. 61
rest, multi-measure ............................ 59, 62
rest, specifying vertical position .......... 59
rest, splitting .................................. 80
rest, style ....................................... 60
rest, whole, for a full measure ............. 62
rest, whole-measure ............................ 59
rest, within text, by log and dot-count ... 746
rest, within text, by string .................. 747
\rest-by-number ............................... 746
\rest-event ...................................... 216
RestEngraver .................................. 81
RestCollision .................................. 182
restNumberThreshold ......................... 224
restoring default properties for time signatures ... 68
\restrainOpenStrings ......................... 357
\retrograde ...................................... 14, 818
retrograde transformation .................... 14
reverb in MIDI .................................. 541
‘reverse turn’ ornament ....................... 122, 768
\reverseTurn ............................... 122, 768
\revert .......................................... 622
reverting override ............................ 622
RevertProperty ................................ 619
\revertTimeSignature ........................ 68, 818
\rfz .............................................. 125
rgb color .......................................... 233
rgb-color ........................................ 233
\rheel ........................................... 122, 769
rhythm, showing melody ...................... 82
rhythmic staff .................................. 191
Rhythmic_column_ engraver ................. 614
RhythmicStaff .................................. 46, 84, 192
RhythmicStaff .................................. 191
Rhythms .................. 49, 54, 55, 59, 61, 62, 66, 71, 75, 77, 80, 81, 84, 87, 94, 98, 99, 107, 112, 113, 115, 119, 121
ride bell ........................................ 770
ride cymbal ..................................... 770
\right-align ..................................... 257, 762
right-aligning text ........................... 732
\right-brace ........................................ 762
\right-column ................................... 732
right-hand fingering, for fretted instrument ........ 396
right-hand fingering, placement ................... 397
right-margin ...................................... 552
\rightHandFinger .................................. 396, 818
\roman ............................................. 716
\romanStringNumbers ............................. 351, 355
root of chord ..................................... 425
\rotate ........................................... 732
rotating object ................................... 649
rotating text ..................................... 732
\rounded-box ..................................... 202, 743
\rtoe ............................................. 122, 769

S
s ..................................................... 61
Sacred Harp note head ............................ 42
\accredHarpHeads ................................. 42
\accredHarpHeadsMinor ......................... 42
\sans ............................................. 716
SATB ............................................. 314
scalable vector graphics output ................. 528
\scale ........................................... 743
\scaleDurations .................................. 54, 77, 818
scaling duration ................................ 54
scaling markup ................................... 743
scaling text ...................................... 733
Scheme object ................................... 775
Scheme tutorial .................................. 597
Scheme variable .................................. 775
Scheme, pure container ........................... 661
Scheme, unpure container ....................... 661
\scoreatur ....................................... 24
\score ............................................ 122, 776, 780
Score is a (single) compound ................. 486, 490, 747
\score ............................................ 486, 490, 747
\scoreTitleMarkup ................................. 500
Scottish highland bagpipe ....................... 413
Script ............................................ 123, 125, 461
\script ............................................ 123, 125, 461
\script, on multi-measure rest ................. 63
\script, vertical ordering ....................... 124
\scripted  ....................................... 461
\scriptEvent ..................................... 461
scripts ........................................... 768
seconda volta .................................... 151
section parts .................................... 210
\segno ............................................ 122, 770
'segno' sign ..................................... 101, 114, 122, 770
'segno' sign, on bar line ....................... 248
'segno' sign, with repeats ...................... 154
selecting font size (notation) ................. 225
self-alignment-interface ....................... 617, 650
self-alignment-X ................................ 567
semia ............................................. 483
semia form ....................................... 482
semi-flat ......................................... 6, 9
semi-flat symbol, appearance ................. 480
semi-sharp ........................................ 6, 9
semi-transparent colors ......................... 232
\semicirculus ..................................... 460, 770
'semicirculus' gregorian articulation ........ 460, 770
\semitab .......................................... 748
\semiGermanChords ............................... 432
\semisharp ....................................... 749
separate text ..................................... 251
sesqui-flat ........................................ 9
sesqui-sharp ..................................... 9
\sesquiflat ....................................... 749
\sesquisharp ..................................... 749
\set .............................................. 87, 620, 625
set-global-fonts .................................. 272
set-global-staff-size ............................ 559
setting extent of text object .................. 764
setting horizontal text alignment ............ 725
Setting simple songs ............................. 274, 275
setting subscript, in standard font size ........ 714
setting superscript, in standard font size ..... 714
\settingsFrom ..................................... 818
seventh chord .................................... 425
\sf ............................................... 125
\ff ............................................... 125
\ffz ............................................... 125
\shape ........................................... 656, 818
shape note ........................................ 42
shaping slurs and ties ......................... 656
shared property .................................. 774
\sharp ............................................. 7
\sharp ............................................. 6
\sharp ............................................. 749
sharp, double ...................................... 6
shift note ......................................... 178
shift rest, automatic ........................... 178
shift symbol, accordion .......................... 348
\shiftDurations ................................... 193
shifting voice .................................... 178
\shiftOff .......................................... 178
\shiftOn .......................................... 178
\shiftOnnn ....................................... 178
\shiftOnn ......................................... 178
short fermata ..................................... 122, 769
short-indent ...................................... 211, 553
shortened volta brackets ....................... 156
\shortfemtata .................................... 122, 769
show-available-fonts ............................ 271
showFirstLength .................................. 528, 775
\showKeySignature ............................... 413
showLastLength .................................. 528, 775
\showStaffSwitch ................................. 344
side-position-interface ......................... 617, 650
sidestick ......................................... 770
sign, coda ........................................ 114, 122, 770
sign, conducting .................................. 93
sign, 'segno' ..................................... 101, 114, 122, 770
sign, segno, with repeats ....................... 154
sign, snappizzicato ............................. 122, 769
sign, variant coda ................................ 122, 770
signature, polyrhythmic ......................... 77
'signum congruentiae' mensural notation ....... 770
'signum congruentiae' ornament .............. 122
\signumcongruentiae .............................. 122, 770
silenced note, percussion 409
single staff with bracket or brace 194
single-line comment 487, 491
single-staff polyphony 173
Size of objects 205
skip 61
SkipMusic 62
skipping notes in lyrics 299
slashChordSeparator 433
slashChordSeparators 762
slashChordSeparators 45
slashed digit 117
\slashed-digit 762
\slashedGrace 115, 819
\slash Separator 556
\slash cycle 122, 768
slide, in tablature notation 366
slope, ottava spanner, modification 26
slur 136
\slur 136, 148
slur style 133
slur, above notes 134
slur, above notes 158
slur, below notes 134
slur, dashed 134
slur, dashed phrasing 137
slur, defining dash patterns 135
slur, defining dash patterns for phrasing 137
slur, dotted 134
slur, dotted phrasing 137
slur, for tuplets 50
slur, half dashed and half solid 134
slur, half solid and half dashed phrasing 137
slur, manual placement 134
slur, modifying 655
slur, multiple 134
slur, multiple phrasing 136
slur, phrasing 134, 136
slur, phrasing, defining dash patterns 137
slur, simultaneous 134
slur, simultaneous phrasing 136
slur, solid 134
slur, text markup inside 135
slur-event 216
\slurDown 134
\slur HalfDashed 134
\slur HalfSolid 134
\slurNeutral 134
\slur Solid 134
\slurUp 135
\small 225, 256, 717
\smallCaps 717
\smaller 254, 256, 717
smallest note 217
smaller note 217
solesmes 447
solid slur 134
solid text string 183
solo part 183
Songs 276, 314
soprano clef 17, 706
sos 347
sostenuto pedal 347
SostenutoEvent 347
\sostenutoOff 347
\sostenutoOn 347
SostenutoPedal 347
SostenutoPedalLineSpanner 347
sound 531
\sourcefileline 491
\sourcefilename 491
Solesmes 447
spatial grouping 42
spatial grouping 42
spatial grouping 43
\sp 125
space, between staves 567
space, in lyrics 273, 282
space, inside systems 567
space, note, spacing 61
space, rest, spacing 61
Spacing 549, 551, 553, 556, 558, 560, 563, 564, 565, 566, 571, 572, 573, 581, 582, 583, 584, 587, 593, 594, 595, 596
spacing 583
spacing, horizontal 582
spacing, vertical 566
spacing-spanner-interface 804, 806
SpacingSpanner 582, 583, 584
\spacingTweaks 819
span bar 104
\span_bar 104
Span_stem_engraver 345
SpanBar 107
spanner, modifying 659
special arpeggio symbol 145
special character 524
special character, in markup mode 253
special note head 39
split into tagged music 523
splitting notes 80
splitting rests 80
Sprechgesang 324
Appendix E: LilyPond index

898

square bracket, at start of staff group ........... 194
square neumes ligature ........................... 461
\staccatissimo ........................................ 123
\startGroup ............................................ 122, 768
'staccatissimo' articulation ........................ 122, 768
\startStaff .............................................. 199, 202
\startStaffSpan ......................................... 148
\staves .................................................. 192
staves, divisi .......................................... 208
staff ...................................................... 192, 202, 205
Staff ...(35, 38, 80, 192, 196, 210, 213, 242, 583, 776
staff change, manual ................................ 342
staff change, manual ................................ 340
staff distance ......................................... 567
staff group .............................................. 193
staff group, with square bracket at start ........ 194
staff initiation ......................................... 191
staff instantiation ...................................... 191
staff line, modifying .................................. 199
staff line, stopping and starting ................... 199
Staff notation ...(74, 192, 196, 197, 198, 202, 205, 210,
... 213, 217, 222
staff size, setting ..................................... 559
staff switching ......................................... 344
staff symbol ............................................ 199
staff symbol, setting of .............................. 636
staff, choir .............................................. 193
staff, drum .............................................. 191
staff, empty ............................................ 205
staff, Frenched ......................................... 202
staff, grand ............................................. 193
staff, hiding ............................................ 205
staff, keyboard instruments ......................... 339
staff, keyed instruments .............................. 339
staff, metronome mark below ......................... 73
staff, multiple ......................................... 193
staff, nested ............................................ 196
staff, new ............................................... 191
staff, percussion ....................................... 191
staff, piano ............................................. 193, 339
staff, resizing of ...................................... 202
staff, single ............................................ 191
staff, single, with bracket or brace ................ 194
staff-affinity ............................................ 567
staff-change line ...................................... 344
\staff .................................................... 230
\staff-space ............................................ 635
\staff-staff-spacing ................................ 567
\staff-symbol-interface .............................. 202
\Staff .................................................. 542
\Staff_c 
\Staff_col 
\Staff_s 
\Staff_ 
\Staff_symbol_ 
StaffGroup ............................................... 112, 196, 197
staffgroup-staff-spacing ............................ 567
StaffGroup .............................................. 316, 568, 570, 572, 631
StaffSpacing ............................................ 583
StaffSymbol ............................................ 192, 202, 205
standalone text ....................................... 251
standard font size (notation) ....................... 228
stanzas number ....................................... 306
StanzaNumber .......................................... 313
start of system ........................................ 193
start repeat ............................................ 159
\start-repeat ........................................... 159
\startAcciaccaturaMusic ................................ 118
\startAppoggiaturaMusic ................................ 118
\startGraceMusic ....................................... 118
\startGroup ............................................ 240
\startStaff ............................................. 199, 202
\startStaffSpan ........................................ 148
\staves .................................................. 192
staves, divisi .......................................... 208
stem ...................................................... 235, 346, 627
stem, automatic direction on center line .......... 235
stem, cross-staff ...................................... 345
stem, default direction on center line ............ 235
stem, direction ........................................ 234
stem, down .............................................. 234
stem, in tablature ..................................... 362
stem, invisible ......................................... 234
stem, neutral .......................................... 234
stem, up .................................................. 234
stem, with slash ....................................... 117
stem-interface ......................................... 235
stem-spacing-correction ............................ 583
Stem ..................................................... 98, 235
\stemDown .............................................. 234
\stemLeftBeamCount .................................. 96, 97
\stemNeutral .......................................... 234
\stemRightBeamCount ................................ 96, 97
\stemUp .................................................. 234
stencil ................................................... 775
\stencil ................................................. 763
stencil, removing ...................................... 642
stereo balance in MIDI ................................ 541
stopAcciaccaturaMusic ................................ 118
stopAppoggiaturaMusic .............................. 118
\stopGraceMusic ....................................... 118
\stopGroup .............................................. 240
\stopped ............................................... 122, 407, 411, 769
'stopped' articulation ................................ 122, 411, 769
\stopStaff .............................................. 199, 202, 205
\stopTrillSpan ......................................... 148
\storePredefinedDiagram ............................. 387, 392, 819
strict spacing and grace notes ...................... 118
strict-beat beaming ................................... 92
strictBeatBeaming ..................................... 92
string ..................................................... 351
string number ......................................... 351
string numbers, orientation ......................... 230
String quartet templates ............................ 351
string vs. fingering number ......................... 354
string, indicating open ................................ 351
StringNumber .......................................... 355
stringNumberOrientations ............................ 230
strings, orchestral .................................... 350
strings, writing for ................................... 350
\stringTuning .......................................... 370, 819
stringTunings ......................................... 370, 383
stroke finger, orientation ........................... 230
\StrokeFinger .......................................... 398
strokeFingerOrientations ............................ 230, 397
\strophes .............................................. 461, 468
Structure of a note entry ............................ 768
Appendix E: LilyPond index

Style sheets .................................. 524
style, bar number .................................. 109
style, double repeat for volta ...................... 157
style, measure number .................................. 109
style, multi-measure rests .......................... 224
style, note heads .................................. 39
style, rehearsal mark .................................. 114
style, rests .................................. 60
style, slur .................................. 134
style, text dynamics .................................. 130
style, voice .................................. 177
\stretchedNoteHeads .................................. 819
subscript .................................. 255, 718
subdivisionBeams .................................. 91
subdividing beams .................................. 91
subscript text .................................. 718
suggestAccidentals .................................. 124, 455
\super .................................. 255, 718
superscript .................................. 255
superscript text .................................. 718
suppressing repeated chords ......................... 391, 431
sus .................................. 427
sustain pedal .................................. 347
sustain pedal style .................................. 347
SustainEvent .................................. 234, 437
\sustainOff .................................. 347
\sustainOn .................................. 347
SustainPedal .................................. 347
SustainPedalLineSpanner .................................. 347
SVG output .................................. 528
swing script .................................. 534
swing.ly .................................. 543
switching font .................................. 254
syllable duration, automatic ......................... 278
symbol, breath mark, changing ...................... 138
symbol, major seven .................................. 434
symbol, non-musical .................................. 263
syntax, markup .................................. 252
system .................................. 192
system separator mark .................................. 198
system start delimiter .................................. 193
system start delimiter, nested ...................... 196
system-count .................................. 554
system-separator-markup ____________________________ 556
system-system-spacing .................................. 551
systems-per-page .................................. 554
SystemStartBar .................................. 196, 197
SystemStartBrace .................................. 196, 197
SystemStartBracket .................................. 196, 197
SystemStartSquare .................................. 196, 197
tablature .................................. 191, 354
tablature, and beams .................................. 362
tablature, and harmonic indications ................. 360
tablature, and harmonics .................................. 363
tablature, and polyphony .................................. 362
tablature, and slides .................................. 366
tablature, and stems .................................. 362
tablature, banjo .................................. 354, 370, 401
tablature, basic .................................. 355
tablature, bass .................................. 370
tablature, bass guitar .................................. 370
tablature, cello .................................. 370
tablature, chord glissando .................................. 367
tablature, custom .................................. 370
tablature, custom string tunings ................. 357
tablature, default .................................. 355
tablature, double bass .................................. 370
tablature, guitar .................................. 354, 370
tablature, hammer on .................................. 368
tablature, lute .................................. 401
tablature, mandolin .................................. 370
tablature, predefined string tunings ............ 370
tablature, pull off .................................. 368
tablature, ukulele .................................. 370
tablature, viola .................................. 370
tablature, violin .................................. 370
\table .................................. 516
table of contents .................................. 514
table of contents, customized functions ........... 516
\tableofcontents .................................. 517, 766
TabNoteHead .................................. 369
TabStaff .................................. 192, 369
TabStaff .................................. 191, 355
TabVoice .................................. 369
TabVoice .................................. 355
tag .................................. 520
\tag .................................. 520, 819
tag groups .................................. 523\tagGroup .................................. 523, 819
tam tam .................................. 770
tambourine .................................. 770\taor .................................. 413
tagasim .................................. 483
tagasim .................................. 482
teaching .................................. 34
teaching accidental style .................................. 34\text .................................. 225, 256, 718
template, Arabic music .................................. 483
tempo .................................. 71\tempo .................................. 71
tempo indication .................................. 74\tempoChange .................................. 813\temporay .................................. 628, 660, 819
tenor clef .................................. 17, 706
tenor clef, choral .................................. 17
tenor G clef .................................. 706
tenor varC clef .................................. 706
tenuto .................................. 125\tenuto .................................. 122, 768
‘tenuto’ articulation .................................. 122, 768
Text .................................. 245, 246, 248, 251, 252, 254, 257, 262, 264, 266, 267, 271, 276
text .................................. 347, 718
text alignment command .................................. 261
<table>
<thead>
<tr>
<th>Text</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tie, laissez vibre</td>
<td>56</td>
</tr>
<tr>
<td>tie, manual engraving</td>
<td>58</td>
</tr>
<tr>
<td>tie, modifying</td>
<td>655</td>
</tr>
<tr>
<td>tie, placement</td>
<td>57</td>
</tr>
<tr>
<td>tie, repeating</td>
<td>56</td>
</tr>
<tr>
<td>tie, with arpeggios</td>
<td>58</td>
</tr>
<tr>
<td>tie-ing text</td>
<td>719</td>
</tr>
<tr>
<td>TieColumn</td>
<td>59, 659</td>
</tr>
<tr>
<td>TieColumn</td>
<td>58</td>
</tr>
<tr>
<td>TieDashed</td>
<td>57</td>
</tr>
<tr>
<td>tieDashed</td>
<td>57</td>
</tr>
<tr>
<td>tieDown</td>
<td>57</td>
</tr>
<tr>
<td>\tied-lyric</td>
<td>749</td>
</tr>
<tr>
<td>\tieDashed</td>
<td>57</td>
</tr>
<tr>
<td>\tieDashPattern</td>
<td>57, 860</td>
</tr>
<tr>
<td>\tieDotted</td>
<td>57</td>
</tr>
<tr>
<td>\tieDashed</td>
<td>57</td>
</tr>
<tr>
<td>\tieHalfDashed</td>
<td>57</td>
</tr>
<tr>
<td>\tieHalfSolid</td>
<td>57</td>
</tr>
<tr>
<td>\tieNeutral</td>
<td>57</td>
</tr>
<tr>
<td>\tieSolid</td>
<td>57</td>
</tr>
<tr>
<td>\tieUp</td>
<td>57</td>
</tr>
<tr>
<td>tieWaitForNote</td>
<td>58</td>
</tr>
<tr>
<td>\time</td>
<td>66, 87, 820</td>
</tr>
<tr>
<td>time administration</td>
<td>121</td>
</tr>
<tr>
<td>time signature</td>
<td>71</td>
</tr>
<tr>
<td>time signature</td>
<td>66</td>
</tr>
<tr>
<td>time signature style</td>
<td>67, 452</td>
</tr>
<tr>
<td>time signature, compound</td>
<td>79</td>
</tr>
<tr>
<td>time signature, default settings</td>
<td>67</td>
</tr>
<tr>
<td>time signature, double</td>
<td>77</td>
</tr>
<tr>
<td>time signature, mensural</td>
<td>452</td>
</tr>
<tr>
<td>time signature, mid-measure</td>
<td>74</td>
</tr>
<tr>
<td>time signature, multiple</td>
<td>606</td>
</tr>
<tr>
<td>time signature, polymeric</td>
<td>77</td>
</tr>
<tr>
<td>time signature, printing only numerator</td>
<td>70</td>
</tr>
<tr>
<td>time signature, properties, restoring default values</td>
<td>68</td>
</tr>
<tr>
<td>time signature, visibility</td>
<td>67</td>
</tr>
<tr>
<td>times</td>
<td>66, 820</td>
</tr>
<tr>
<td>TimeScaledMusic</td>
<td>54</td>
</tr>
<tr>
<td>TimeSignature</td>
<td>71, 80</td>
</tr>
<tr>
<td>timeSignatureFraction</td>
<td>77</td>
</tr>
<tr>
<td>timing information and repeats</td>
<td>158</td>
</tr>
<tr>
<td>timing mark, for glissando</td>
<td>142</td>
</tr>
<tr>
<td>timing, within score</td>
<td>121</td>
</tr>
<tr>
<td>Timing_translator</td>
<td>71, 75, 80, 107, 122, 780</td>
</tr>
<tr>
<td>\tiny</td>
<td>225, 256, 719</td>
</tr>
<tr>
<td>title</td>
<td>492</td>
</tr>
<tr>
<td>toc</td>
<td>514</td>
</tr>
<tr>
<td>tocFormatMarkup</td>
<td>517</td>
</tr>
<tr>
<td>tocIndentMarkup</td>
<td>517</td>
</tr>
<tr>
<td>tocItem</td>
<td>517, 820</td>
</tr>
<tr>
<td>tocItemWithDotsMarkup</td>
<td>516</td>
</tr>
<tr>
<td>tocTitleMarkup</td>
<td>517</td>
</tr>
<tr>
<td>`toe' pedal mark</td>
<td>122, 769</td>
</tr>
<tr>
<td>tom tom</td>
<td>770</td>
</tr>
<tr>
<td>Top</td>
<td>1, 597</td>
</tr>
<tr>
<td>top-level text</td>
<td>251</td>
</tr>
<tr>
<td>top-margin</td>
<td>549</td>
</tr>
<tr>
<td>top-markup-spacing</td>
<td>551</td>
</tr>
<tr>
<td>top-system-spacing</td>
<td>551</td>
</tr>
<tr>
<td>toplevel-bookparts</td>
<td>775</td>
</tr>
<tr>
<td>toplevel-scores</td>
<td>775</td>
</tr>
<tr>
<td>transcription of mensural music</td>
<td>195</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Voice</td>
<td>29, 30</td>
</tr>
<tr>
<td>voice accidental style</td>
<td>30</td>
</tr>
<tr>
<td>voice style</td>
<td>177</td>
</tr>
<tr>
<td>voice, additional, in polyphonic music</td>
<td>181</td>
</tr>
<tr>
<td>voice, ambitus</td>
<td>36</td>
</tr>
<tr>
<td>voice, divided</td>
<td>317</td>
</tr>
<tr>
<td>voice, extra, for handling breaks</td>
<td>562</td>
</tr>
<tr>
<td>voice, following</td>
<td>344</td>
</tr>
<tr>
<td>voice, multiple</td>
<td>178</td>
</tr>
<tr>
<td>voice, \partCombine with \autoBeamOff</td>
<td>86</td>
</tr>
<tr>
<td>voice, quoting</td>
<td>214, 217</td>
</tr>
<tr>
<td>voice, shifting</td>
<td>178</td>
</tr>
<tr>
<td>voice, with ottavation</td>
<td>26</td>
</tr>
<tr>
<td>VoiceFollower</td>
<td>345, 641</td>
</tr>
<tr>
<td>voiceFour</td>
<td>173</td>
</tr>
<tr>
<td>voiceFourStyle</td>
<td>177</td>
</tr>
<tr>
<td>\voiceNeutralStyle</td>
<td>177</td>
</tr>
<tr>
<td>\voiceOne</td>
<td>173</td>
</tr>
<tr>
<td>\voiceOneStyle</td>
<td>177</td>
</tr>
<tr>
<td>voices</td>
<td>176, 821</td>
</tr>
<tr>
<td>Voices contain music</td>
<td>177, 182</td>
</tr>
<tr>
<td>\voiceThree</td>
<td>173</td>
</tr>
<tr>
<td>\voiceThreeStyle</td>
<td>177</td>
</tr>
<tr>
<td>\voiceTwo</td>
<td>173</td>
</tr>
<tr>
<td>\voiceTwoStyle</td>
<td>177</td>
</tr>
<tr>
<td>\void</td>
<td>544, 821</td>
</tr>
<tr>
<td>volta</td>
<td>158</td>
</tr>
<tr>
<td>volta</td>
<td>151</td>
</tr>
<tr>
<td>volta bracket</td>
<td>159</td>
</tr>
<tr>
<td>volta bracket, and tie</td>
<td>56</td>
</tr>
<tr>
<td>volta bracket, in additional staves</td>
<td>156</td>
</tr>
<tr>
<td>volta bracket, shortened</td>
<td>156</td>
</tr>
<tr>
<td>volta bracket, with text</td>
<td>160</td>
</tr>
<tr>
<td>volta repeat, below chords</td>
<td>435</td>
</tr>
<tr>
<td>volta, double repeat style</td>
<td>157</td>
</tr>
<tr>
<td>volta, prima</td>
<td>151</td>
</tr>
<tr>
<td>volta, seconda</td>
<td>151</td>
</tr>
<tr>
<td>Volta_bracket</td>
<td>156</td>
</tr>
<tr>
<td>\VoltaBracket</td>
<td>158, 161</td>
</tr>
<tr>
<td>VoltaRepeatedMusic</td>
<td>158, 161</td>
</tr>
<tr>
<td>vowel transition</td>
<td>287</td>
</tr>
<tr>
<td>vowel transition, repeat</td>
<td>286</td>
</tr>
<tr>
<td>VowelTransition</td>
<td>287</td>
</tr>
<tr>
<td>vspace</td>
<td>259, 733</td>
</tr>
</tbody>
</table>

**W**

Walker shape note head                                     42
\walkerHeads                                               42
\walkerHeadsMinor                                         43
\whichBar                                                  106
whistle                                                   770
white mensural ligature                                   456
\whitout                                                   763
whitespace                                               491
\whiteTriangleMarkup                                      432
whole rest, for a full measure                          62
wind instrument                                          410
wind instruments, fingering symbols                      411
Winds                                                       411, 413, 414, 415, 423
\with                                                      665, 680
\with-color                                               232, 764
Within-staff objects ........................................ 635
\withMusicProperty ........................................ 821
woodblock .................................................. 770
woodwind diagram, key lists ............................... 422
woodwind diagram, list ..................................... 418
woodwind diagrams, modifying ............................. 421
\woodwind-diagram ......................................... 752
\wordwrap .................................................. 261, 735
\wordwrap-field ............................................ 734
\wordwrap-internal ......................................... 766
\wordwrap-lines ............................................ 207, 766
\wordwrap-string ........................................... 735
\wordwrap-string-internal ................................. 767
\wordwrapped text .......................................... 261
Working on input files .................................... 487
World music .................................................. 479, 480, 482, 483
writing music in parallel ................................. 188
written-out repeat ......................................... 161
X
x11 color .................................................... 232, 233
x11-color ..................................................... 232, 233
X-offset ...................................................... 567
\xNote ......................................................... 39, 821
\xNotesOff .................................................. 39
\xNotesOn .................................................... 39