

Live Object Model

Version 9.2.0-rev.1

Cycling ’74

cycling74.com

Contents

LOM - The Live Object Model	4
Application.View	8
Application	12
Chain	16
ChainMixerDevice	21
Clip.View	23
Clip	25
ClipSlot	51
CompressorDevice	57
ControlSurface	59
CuePoint	63
Device.View	65
Device	67
DeviceIO	72
DeviceParameter	74
DriftDevice	79
DrumCellDevice	86
DrumChain	87
DrumPad	89
Eq8Device.View	91
Eq8Device	92
Groove	94
GroovePool	97
HybridReverbDevice	98
LooperDevice	101
MaxDevice	105
MeldDevice	108
MixerDevice	110
PluginDevice	113
RackDevice.View	115
RackDevice	117
RoarDevice	123
Sample	125
Scene	133
ShifterDevice	138

SimplerDevice.View 139
SimplerDevice 140
Song.View 145
Song 149
SpectralResonatorDevice 170
TakeLane 173
this_device 175
Track.View 176
Track 178
TuningSystem 194
WavetableDevice 196
Credits 202

LOM - The Live Object Model

Object Model Overview 0
API Objects 4

Objects which comprise the Live API described by their structure, properties and functions. The Live Object Model lists a number of Live object classes with their properties and functions, as well as their parent-child relations through which a hierarchy is formed. Please refer to the Live API overview chapterfor definitions of the basic Live API terms and a list of the Max objects used to access it.

This document refers to Ableton Live version 12.4.5

API Objects

Item	Description
Application	This class represents the Live application. It is reachable by the root path live_app ...
Application.View	This class represents the aspects of the Live application related to viewing the application....
Chain	This class represents a group device chain in Live.
ChainMixerDevice	This class represents a chain's mixer device in Live.
Clip	This class represents a clip in Live. It can be either an audio clip or a MIDI clip in the Arr...
Clip.View	Representing the view aspects of a Clip.
ClipSlot	This class represents an entry in Live's Session View matrix. The properties ...

CompressorDevice	This class represents a Compressor device in Live. A CompressorDevice shares all of the ch...
ControlSurface	A ControlSurface can be reached either directly by the root path control_surfaces N or by g...
CuePoint	Represents a locator in the Arrangement View.
Device	This class represents a MIDI or audio device in Live.
Device.View	Representing the view aspects of a Device.
DeviceIO	This class represents an input or output bus of a Live device.
DeviceParameter	This class represents an (automatable) parameter within a MIDI or audio device. To modify a de...
DriftDevice	This class represents an instance of a Drift device in Live. A DriftDevice has all the...
DrumCellDevice	This class represents an instance of a Drum Sampler device in Live. A DrumCell has all...
DrumChain	This class represents a Drum Rack device chain in Live. A DrumChain is a type ...
DrumPad	This class represents a Drum Rack pad in Live.
Eq8Device	This class represents an instance of an EQ Eight device in Live. An Eq8Device has all ...
Eq8Device.View	Represents the view aspects of an Eq8Device. An Eq8Device.View has all the children, p...
Groove	This class represents a groove in Live. Available since Live 11.0. ...
GroovePool	This class represents the groove pool in Live. It provides access to the current set's list of groov...
HybridReverbDevice	This class represents an instance of a Hybrid Reverb device in Live. A HybridReverbDev...
LooperDevice	This class represents an instance of a Looper device in Live. An LooperDevice has all ...

MaxDevice	This class represents a Max for Live device in Live. A MaxDevice is a type of Device...
MeldDevice	This class represents an instance of a Meld device in Live. A MeldDevice has all the p...
MixerDevice	This class represents a mixer device in Live. It provides access to volume, panning and other ...
PluginDevice	This class represents a plug-in device. A PluginDevice is a type of Device, meaning ...
RackDevice	This class represents a Live Rack Device. A RackDevice is a type of Device, meaning th...
RackDevice.View	Represents the view aspects of a Rack Device. A RackDevice.View is a type of Device.Vi...
RoarDevice	This class represents an instance of a Roar device in Live. A RoarDevice has all the p...
Sample	This class represents a sample file loaded into Simpler.
Scene	This class represents a series of clip slots in Live's Session View matrix....
ShifterDevice	This class represents an instance of the Shifter audio effect. A ShifterDevice is a ty...
SimplerDevice	This class represents an instance of Simpler. A SimplerDevice is a type of device, mea...
SimplerDevice.View	Represents the view aspects of a SimplerDevice. A SimplerDevice.View is a type of Device.V...
Song	This class represents a Live Set. The current Live Set is reachable by the root path li...
Song.View	This class represents the view aspects of a Live document: the Session and Arrangement Views....
SpectralResonatorDevice	This class represents an instance of a Spectral Resonator device in Live. An SpectralR...

TakeLane	This class represents a take lane in Live. Tracks in Live can have take lanes in Arrangement View, w...
this_device	This root path represents the device containing the live.path object to which the ...
Track	This class represents a track in Live. It can either be an audio track, a MIDI track, a return...
Track.View	Representing the view aspects of a track.
TuningSystem	This class represents a tuning system in Live.
WavetableDevice	This class represents a Wavetable instrument. A WavetableDevice shares all of the ch...

Application.View

Canonical Path 8
Properties 8
  browse_mode 8
  focused_document_view 9
Functions 9
  available_main_views 9
  focus_view 9
  hide_view 9
  is_view_visible 9
  scroll_view 10
  show_view 10
  toggle_browse 10
  zoom_view 10

This class represents the aspects of the Live application related to viewing the application.

Canonical Path

```
live_app view
```

Properties

browse_mode bool

Access: get observe

1 = Hot-Swap Mode is active for any target.

focused_document_view unicode

Access: get observe

The name of the currently visible view in the focused Live window ('Session' or 'Arranger').

Functions

available_main_views

Returns: `view names` [list of symbols].
This is a constant list of view names to be used as an argument when calling other functions: `Browser Arranger Session Detail Detail/Clip Detail/DeviceChain`.

focus_view

Parameter: `view_name`
Shows named view and focuses on it. You can also pass an empty view_name “ ", which refers to the Arrangement or Session View (whichever is visible in the main window).

hide_view

Parameter: `view_name`
Hides the named view. You can also pass an empty view_name “ ", which refers to the Arrangement or Session View (whichever is visible in the main window).

is_view_visible

Parameter: `view_name`
Returns: [bool] Whether the specified view is currently visible.

scroll_view

Parameters: `direction view_name modifier_pressed`
`direction` [int] is 0 = up, 1 = down, 2 = left, 3 = right
`modifier_pressed` [bool] If view_name is "Arranger" and modifier_pressed is 1 and direction is left or right, then the size of the selected time region is modified, otherwise the position of the playback cursor is moved.
Not all views are scrollable, and not in all directions. Currently, only the `Arranger`, `Browser`, `Session`, and `Detail/DeviceChain` views can be scrolled.
You can also pass an empty view_name `" "`, which refers to the Arrangement or Session View (whichever view is visible).

show_view

Parameter: `view_name`

toggle_browse

Displays the device chain and the browser and activates Hot-Swap Mode for the selected device. Calling this function again deactivates Hot-Swap Mode.

zoom_view

Parameter: `direction view_name modifier_pressed`
`direction` [int] - 0 = up, 1 = down, 2 = left, 3 = right
`modifier_pressed` [bool] If `view_name` is 'Arrangement', `modifier_pressed` is 1, and `direction` is left or right, then the size of the selected time region is modified, otherwise the position of the playback cursor is moved. If `view_name` is Arrangement and `modifier_pressed` is 1 and `direction` is up or down, then only the height of the highlighted track is changed, otherwise the height of all tracks is changed.
Only the Arrangement and Session Views can be zoomed. For Session View, the behaviour of

zoom_view is identical to scroll_view. You can also pass an empty view_name “ ", which refers to the Arrangement or Session View (whichever is visible in the main window).

Application

Canonical Path 12
Children 12
  view 13
  control_surfaces 13
Properties 13
  current_dialog_button_count 13
  current_dialog_message 13
  open_dialog_count 13
  average_process_usage 14
  peak_process_usage 14
Functions 14
  get_bugfix_version 14
  get_document 14
  get_major_version 14
  get_minor_version 15
  get_version_string 15
  press_current_dialog_button 15

This class represents the Live application. It is reachable by the root path `live_app`.

Canonical Path

```
live_app
```

Children

view Application.View

Access: get

control_surfaces list of ControlSurface

Access: get observe

A list of the control surfaces currently selected in Live's Preferences.
If None is selected in any of the slots or the script is inactive (e.g. when Push2 is selected, but no Push is connected), id 0 will be returned at those indices.

Properties

current_dialog_button_count int

Access: get

The number of buttons in the current message box.

current_dialog_message symbol

Access: get

The text of the current message box (empty if no message box is currently shown).

open_dialog_count int

Access: get observe

The number of dialog boxes shown.

average_process_usage float

Access: get observe

Reports Live's average CPU load.

Note that Live's CPU meter shows the audio processing load but not Live's overall CPU usage.

peak_process_usage float

Access: get observe

Reports Live's peak CPU load.

Note that Live's CPU meter shows the audio processing load but not Live's overall CPU usage.

Functions

get_bugfix_version

Returns: the 2 in Live 9.1.2.

get_document

Returns: the current Live Set.

get_major_version

Returns: the 9 in Live 9.1.2.

get_minor_version

Returns: the 1 in Live 9.1.2.

get_version_string

Returns: the text 9.1.2 in Live 9.1.2.

press_current_dialog_button

Parameter: `index`
Press the button with the given index in the current dialog box.

Chain

Canonical Paths 16
Children 17
  devices 17
  mixer_device 17
Properties 17
  color 17
  color_index 18
  is_auto_colored 18
  has_audio_input 18
  has_audio_output 18
  has_midi_input 18
  has_midi_output 18
  mute 19
  muted_via_solo 19
  name 19
  solo 19
Functions 19
  delete_device 19
  insert_device 20

This class represents a group device chain in Live.

Canonical Paths

```
live_set tracks N devices M chains L
```

```
live_set tracks N devices M return_chains L
```

```
live_set tracks N devices M chains L devices K chains P ...
```

```
live_set tracks N devices M return_chains L devices K chains P ...
```

Children

devices Device

Access: get observe

mixer_device ChainMixerDevice

Access: get

Properties

color int

Access: get set observe

The RGB value of the chain's color in the form `0x00rrggbb` or (2^16 * red) + (2^8) * green + blue, where red, green and blue are values from 0 (dark) to 255 (light).

When setting the RGB value, the nearest color from the color chooser is taken.

color_index long

Access: get set observe

The color index of the chain.

is_auto_colored bool

Access: get set observe

1 = the chain will always have the color of the containing track or chain.

has_audio_input bool

Access: get

has_audio_output bool

Access: get

has_midi_input bool

Access: get

has_midi_output bool

Access: get

mute bool

Access: get set observe

1 = muted (Chain Activator off)

muted_via_solo bool

Access: get observe

1 = muted due to another chain being soloed.

name unicode

Access: get set observe

solo bool

Access: get set observe

1 = soloed (Solo switch on)
does not automatically turn Solo off in other chains.

Functions

delete_device

Parameter: `index` [int]
Delete the device at the given index.

insert_device

Parameters: `device_name` [symbol] `target_index` [int] (optional)

Attempts to insert the device specified by `device_name` at the given index in the chain. If no index is provided, attempts to insert the device at the end. Throws an error if insertion is not possible. `device_name` is the name as it appears in the UI of Live.
Not all indices are valid. As can be expected, indices outside of the range defined by the current length of the device chain are invalid, but there are other limitations: for example, a MIDI effect can't be inserted after an instrument. The rule of thumb is that if an index would be invalid when inserting using the mouse, it's invalid here.

At the moment, only native Live devices can be inserted. Max for Live devices and plug-in are not supported.

Available since Live 12.3.

ChainMixerDevice

Canonical Paths 21
Children 21
sends 21
chain_activator 22
panning 22
volume 22

This class represents a chain's mixer device in Live.

Canonical Paths

```
live_set tracks N devices M chains L mixer_device
```

```
live_set tracks N devices M return_chains L mixer_device
```

Children

sends list of DeviceParameter

Access: get observe

[in Audio Effect Racks and Instrument Racks only]
For Drum Racks, otherwise empty.

chain_activator DeviceParameter

Access: `get`

panning DeviceParameter

Access: `get`

[in Audio Effect Racks and Instrument Racks only]

volume DeviceParameter

Access: `get`

[in Audio Effect Racks and Instrument Racks only]

Clip.View

Canonical Path 23
Properties 23
grid_is_triplet 23
grid_quantization 24
Functions 24
hide_envelope 24
select_envelope_parameter 24
show_envelope 24
show_loop 24

Representing the view aspects of a Clip.

Canonical Path

```
live_set tracks N clip_slots M clip view
```

Properties

grid_is_triplet bool

Access: get set

Get/set whether the clip is displayed with a triplet grid.

grid_quantization int

Access: get set

Get/set the grid quantization.

Functions

hide_envelope

Hide the Envelopes box.

select_envelope_parameter

Parameter: [DeviceParameter]
Select the specified device parameter in the Envelopes box.

show_envelope

Show the Envelopes box.

show_loop

If the clip is visible in Live's Detail View, this function will make the current loop visible there.

Clip

Canonical Paths 27
Children 27
  view 27
Properties 27
  available_warp_modes 27
  color 28
  color_index 28
  end_marker 28
  end_time 28
  gain 29
  gain_display_string 29
  file_path 29
  groove 29
  has_envelopes 29
  has_groove 30
  is_session_clip 30
  is_arrangement_clip 30
  is_take_lane_clip 30
  is_audio_clip 31
  is_midi_clip 31
  is_overdubbing 31
  is_playing 31
  is_recording 31
  is_triggered 32
  launch_mode 32
  launch_quantization 32
  legato 33
  length 33
  loop_end 33
  loop_jump 33
  loop_start 34
  looping 34
  muted 34
  name 34
  notes 34

warp_markers 35
pitch_coarse 35
pitch_fine 36
playing_position 36
playing_status 36
position 36
ram_mode 37
sample_length 37
sample_rate 37
signature_denominator 37
signature_numerator 37
start_marker 37
start_time 38
velocity_amount 38
warp_mode 38
warping 39
will_record_on_start 39
Functions 39
add_new_notes 39
add_warp_marker 40
apply_note_modifications 41
clear_all_envelopes 41
clear_envelope 41
crop 41
deselect_all_notes 42
duplicate_loop 42
duplicate_notes_by_id 42
duplicate_region 43
fire 43
get_all_notes_extended 43
get_notes_by_id 44
get_notes_extended 45
get_selected_notes_extended 46
move_playing_pos 47
move_warp_marker 47
quantize 47
quantize_pitch 48
remove_notes_by_id 48
remove_notes_extended 48
remove_warp_marker 49
scrub 49

select_all_notes 49
select_notes_by_id 49
set_fire_button_state 49
stop 50
stop_scrub 50

This class represents a clip in Live. It can be either an audio clip or a MIDI clip in the Arrangement or Session View, depending on the track / slot it lives in.

Canonical Paths

```
live_set tracks N clip_slots M clip
```

```
live_set tracks N arrangement_clips M
```

Children

view Clip.View

Access: get

Properties

available_warp_modes list

Access: get

Returns the list of indexes of the Warp Modes available for the clip. Only valid for audio clips.

color int

Access: get set observe

The RGB value of the clip's color in the form `0x00rrggbb` or (2^16 * red) + (2^8) * green + blue, where red, green and blue are values from 0 (dark) to 255 (light).

When setting the RGB value, the nearest color from the clip color chooser is taken.

color_index int

Access: get set observe

The clip's color index.

end_marker float

Access: get set observe

The end marker of the clip in beats, independent of the loop state. Cannot be set before the start marker.

end_time float

Access: get observe

The end time of the clip. For Session View clips, if Loop is on, this is the Loop End, otherwise it's the End Marker. For Arrangement View clips, this is always the position of the clip's rightmost edge in

the Arrangement.

gain float

Access: get set observe

The gain of the clip (range is 0.0 to 1.0). Only valid for audio clips.

gain_display_string symbol

Access: get

Get the gain display value of the clip as a string (e.g. "1.3 dB"). Can only be called on audio clips.

file_path symbol

Access: get

Get the location of the audio file represented by the clip. Only available for audio clips.

groove Groove

Access: get set observe

Get/set/observe access to the groove associated with this clip.

Available since Live 11.0.

has_envelopes bool

Access: get observe

Get/observe whether the clip has any automation.

has_groove bool

Access: get

Returns true if a groove is associated with this clip.

Available since Live 11.0.

is_session_clip bool

Access: get

1 = The clip is a Session clip.
A clip can be either an Arrangement or a Session clip.

is_arrangement_clip bool

Access: get

1 = The clip is an Arrangement clip.
A clip can be either an Arrangement or a Session clip.

is_take_lane_clip bool

Access: get

1 = The clip is a Take Lane clip.
Returns true if the clip is on a Take Lane. Take Lane clips are also Arrangement clips.

is_audio_clip bool

Access: get

0 = MIDI clip, 1 = audio clip

is_midi_clip bool

Access: get

The opposite of `is_audio_clip`.

is_overdubbing bool

Access: get observe

1 = clip is overdubbing.

is_playing bool

Access: get set

1 = clip is playing or recording.

is_recording bool

Access: get observe

1 = clip is recording.

is_triggered bool

Access: get

1 = Clip Launch button is blinking.

launch_mode int

Access: get set observe

The Launch Mode of the Clip as an integer index. Available Launch Modes are:
0 = Trigger (default)
1 = Gate
2 = Toggle
3 = Repeat

Available since Live 11.0.

launch_quantization int

Access: get set observe

The Launch Quantization of the Clip as an integer index. Available Launch Quantization values are:
0 = Global (default)
1 = None
2 = 8 Bars
3 = 4 Bars
4 = 2 Bars
5 = 1 Bar
6 = 1/2
7 = 1/2T
8 = 1/4
9 = 1/4T
10 = 1/8

11 = 1/8T
12 = 1/16
13 = 1/16T
14 = 1/32

Available since Live 11.0.

legato bool

Access: get set observe

1 = Legato Mode switch in the Clip's Launch settings is on.

Available since Live 11.0.

length float

Access: get

For looped clips: loop length in beats. Otherwise it's the distance in beats from start to end marker. Makes no sense for unwarped audio clips.

loop_end float

Access: get set observe

For looped clips: loop end.
For unlooped clips: clip end.

loop_jump bang

Access: observe

Bangs when the clip play position is crossing the loop start marker (possibly projected into the loop).

loop_start float

Access: get set observe

For looped clips: loop start.
For unlooped clips: clip start.

loop_start and loop_end are in absolute clip beat time if clip is MIDI or warped. The 1.1.1 position has beat time 0. If the clip is unwarped audio, they are given in seconds, 0 is the time of the first sample in the audio material.

looping bool

Access: get set observe

1 = clip is looped. Unwarped audio cannot be looped.

muted bool

Access: get set observe

1 = muted (i.e. the Clip Activator button of the clip is off).

name symbol

Access: get set observe

notes bang

Access: observe

Observer sends bang when the list of notes changes.
Available for MIDI clips only.

warp_markers dict/bang

Access: get observe

Observing this property outputs a bang when the Warp Markers change.

Getting this property returns the Warp Markers in a dict as pairs of sample times and beat times:
`sample_time` : [float] the position in seconds in the audio sample file.
`beat_time` : [float] the beat this sample position corresponds to.

To calculate the position in the sample file that corresponds to any given Clip time in beats, Live goes through these steps:\

- Find the Warp Marker with a `beat_time` below the given Clip time and the one above it.
- Get the ratio of the Clip time in beats between the beat times of these two markers.
- Get the `sample_time` for each of these two markers.
- Interpolate between these two sample times with the same ratio to get the file sample position in seconds.
  The last Warp Marker in the dict is not visible in the Live interface. This hidden marker is used to calculate the BPM of the last segment.

  Available for audio clips only.

  Getting is available since Live 11.0.

pitch_coarse int

Access: get set observe

Pitch shift in semitones ("Transpose"), -48 ... 48.
Available for audio clips only.

pitch_fine float

Access: get set observe

Extra pitch shift in cents ("Detune"), -50 ... 49.
Available for audio clips only.

playing_position float

Access: get observe

Current playing position of the clip.

For MIDI and warped audio clips, the value is given in beats of absolute clip time. The clip's beat time of 0 is where 1 is shown in the bar/beat/16th time scale at the top of the clip view.

For unwarped audio clips, the position is given in seconds, according to the time scale shown at the bottom of the clip view.

Stopped clips have a playing position of 0.

playing_status bang

Access: observe

Observer sends bang when playing/trigger status changes.

position float

Access: get observe

Get and set the clip's loop position. The value will always equal loop_start, however setting this property, unlike setting loop_start, preserves the loop length.

ram_mode bool

Access: get set observe

1 = an audio clip’s RAM switch is enabled.

sample_length int

Access: get

Length of the Clip's sample, in samples.

sample_rate float

Access: get

Get the Clip's sample rate.

signature_denominator int

Access: get set observe

signature_numerator int

Access: get set observe

start_marker float

Access: get set observe

The start marker of the clip in beats, independent of the loop state. Cannot be set behind the end marker.

start_time float

Access: get observe

The start time of the clip, relative to the global song time. The value is in beats.

For Arrangement View clips, this is the offset within the arrangement. For Session View clips, this is the time the clip was started. Note that what is reported is the start_time of the currently playing clip on the track, regardless of which clip.

When a Session View clip's playback position was offset by clicking in its time ruler in the Clip Detail View or moving its start marker, its start_time may be negative. This allows using the start_time as an offset when calculating the clip's current playback position based on the global song time.

velocity_amount float

Access: get set observe

How much the velocity of the note that triggers the clip affects its volume, 0 = no effect, 1 = full effect.

Available since Live 11.0.

warp_mode int

Access: get set observe

The Warp Mode of the clip as an integer index. Available Warp Modes are:
0 = Beats Mode
1 = Tones Mode
2 = Texture Mode
3 = Re-Pitch Mode
4 = Complex Mode
5 = REX Mode
6 = Complex Pro Mode
Available for audio clips only.

warping bool

Access: get set observe

1 = Warp switch is on.
Available for audio clips only.

Technical note: Internally, Live will defer the setting of this property. This has the consequence that if you are sequencing API calls from a single event, the actual order of operations may differ from what you'd intuitively expect. Most of the time this should be transparent to you, but if you run into issues, please report them.

will_record_on_start bool

Access: get

1 for MIDI clips which are in triggered state, with the track armed and MIDI Arrangement Overdub on.

Functions

add_new_notes

Parameter:
`dictionary`
Key: `"notes"` [list of note specification dictionaries]
Note specification dictionaries have the following keys:
`pitch` : [int] the MIDI note number, 0...127, 60 is C3.
`start_time` : [float] the note start time in beats of absolute clip time.
`duration` : [float] the note length in beats.
`velocity (optional)` : [float] the note velocity, 0 ... 127 *(100 by default)*.
`mute (optional)` : [bool] 1 = the note is deactivated *(0 by default)*.
`probability (optional)` : [float] the chance that the note will be played:
1.0 = the note is always played
0.0 = the note is never played
(1.0 by default).
`velocity_deviation (optional)` : [float] the range of velocity values at which the note can be played:
0.0 = no deviation; the note will always play at the velocity specified by the *velocity* property
-127.0 to 127.0 = the note will be assigned a velocity value between *velocity* and *velocity + velocity_deviation*, inclusive; if the resulting range exceeds the limits of MIDI velocity (0 to 127), then it will be clamped within those limits
(0.0 by default).
`release_velocity (optional)` : [float] the note release velocity *(64 by default)*.
Returns a list of note IDs of the added notes.

For MIDI clips only.

Available since Live 11.0.

add_warp_marker

Only available for warped Audio Clips. Adds the specified warp marker, if possible.

The warp marker is specified as a dict which can have a `beat_time` and a `sample_time` key, both associated with float values.
The `sample_time` key may be omitted; in this case, Live will calculate the appropriate sample time to create a warp marker at the specified beat time without changing the Clip's playback timing, similar to what would happen if you were to double-click in the upper half of the Sample Display in Clip View.

If `sample_time` is specified, certain limitations must be taken into account: \

- The sample time must lie within the range *[0, s]*, where *s* is the sample's length. The `sample_length` Clip property helps with this.
- The sample time must lie between the left and right adjacents markers' respective sample times (this is a logical constraint).
- Within these constraints, there are limitations on the resulting segments' BPM. The allowed BPM range is *[5, 999]*.

apply_note_modifications

Parameter:
`dictionary`
Key: `"notes"` [list of note dictionaries] as returned from `get_notes_extended`.
The list of note dictionaries passed to the function can be a subset of notes in the clip, but will be ignored if it contains any notes that are not present in the clip.

For MIDI clips only.

Available since Live 11.0. Replaces modifying notes with remove_notes followed by set_notes.

clear_all_envelopes

Removes all automation in the clip.

clear_envelope

Parameter:
`device_parameter` [id]
Removes the automation of the clip for the given parameter.

crop

Crops the clip: if the clip is looped, the region outside the loop is removed; if it isn't, the region outside the start and end markers.

deselect_all_notes

Call this before replace_selected_notes if you just want to add some notes.
Output:
`deselect_all_notes id 0`

For MIDI clips only.

duplicate_loop

Makes the loop two times longer by moving loop_end to the right, and duplicates both the notes and the envelopes. If the clip is not looped, the clip start/end range is duplicated. Available for MIDI clips only.

duplicate_notes_by_id

Parameter:
`list` of note IDs.
Or `dictionary`
Keys:
`note_ids` [list of note IDs] as returned from `get_notes_extended`
`destination_time (optional)` [float/int]
`transposition_amount (optional)` [int]
Duplicates all notes matching the given note IDs.
Provided note IDs must be associated with existing notes in the clip. Existing notes can be queried with `get_notes_extended`.
The selection of notes will be duplicated to *destination_time*, if provided. Otherwise the new notes will be inserted after the last selected note. This behavior can be observed when duplicating notes in the Live GUI.
If the *transposition_amount* is specified, the duplicated notes will be transposed by the number of semitones.
Available for MIDI clips only.

Available since Live 11.1.2

duplicate_region

Parameter:
`region_start` [float/int]
`region_length` [float/int]
`destination_time` [float/int]
`pitch (optional)` [int]
`transposition_amount (optional)` [int]
Duplicate the notes in the specified region to the *destination_time*. Only notes of the specified pitch are duplicated or all if *pitch* is -1. If the *transposition_amount* is not 0, the notes in the region will be transposed by the *transpose_amount* of semitones. Available for MIDI clips only.

fire

Same effect as pressing the Clip Launch button.

get_all_notes_extended

Parameter:
`dict (optional)` [dict]
(See below for a discussion of this argument).

Returns a dictionary of all of the notes in the clip, regardless of where they are positioned with respect to the start/end markers and the loop start/loop end, as a list of note dictionaries. Each note dictionary consists of the following key-value pairs:
`note_id` : [int] the unique note identifier.
`pitch` : [int] the MIDI note number, 0...127, 60 is C3.
`start_time` : [float] the note start time in beats of absolute clip time.
`duration` : [float] the note length in beats.
`velocity` : [float] the note velocity, 0 ... 127.
`mute` : [bool] 1 = the note is deactivated.
`probability` : [float] the chance that the note will be played:
1.0 = the note is always played;

0.0 = the note is never played.
`velocity_deviation` : [float] the range of velocity values at which the note can be played:
0.0 = no deviation; the note will always play at the velocity specified by the *velocity* property
-127.0 to 127.0 = the note will be assigned a velocity value between *velocity* and *velocity + velocity_deviation*, inclusive; if the resulting range exceeds the limits of MIDI velocity (0 to 127), then it will be clamped within those limits.
`release_velocity` : [float] the note release velocity.

It is possible to optionally provide a single [dict] argument to this function, containing a single key-value pair: the key is "return" and the associated value is a list of the note properties as listed above in the discussion of the returned note dictionaries, e.g. ["note_id", "pitch", "velocity"]. The effect of this will be that the returned note dictionaries will only contain the key-value pairs for the specified properties, which can be useful to improve patch performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.1

get_notes_by_id

Parameter:
`list` of note IDs.

Provided note IDs must be associated with existing notes in the clip. Existing notes can be queried with `get_notes_extended`.

Returns a dictionary of notes associated with the provided IDs, as a list of note dictionaries. Each note dictionary consists of the following key-value pairs:
`note_id` : [int] the unique note identifier.
`pitch` : [int] the MIDI note number, 0...127, 60 is C3.
`start_time` : [float] the note start time in beats of absolute clip time.
`duration` : [float] the note length in beats.
`velocity` : [float] the note velocity, 0 ... 127.
`mute` : [bool] 1 = the note is deactivated.
`probability` : [float] the chance that the note will be played:
1.0 = the note is always played;
0.0 = the note is never played.

`velocity_deviation` : [float] the range of velocity values at which the note can be played:
0.0 = no deviation; the note will always play at the velocity specified by the *velocity* property
-127.0 to 127.0 = the note will be assigned a velocity value between *velocity* and *velocity + velocity_deviation*, inclusive; if the resulting range exceeds the limits of MIDI velocity (0 to 127), then it will be clamped within those limits.
`release_velocity` : [float] the note release velocity.

It is possible to optionally provide the argument to this function in the form of a dictionary instead. The dictionary must include the "note_ids" key associated with a list of [int]s, which are the ID values you would like to pass to the function.

If you use this method, you can optionally provide an additional key-value pair: the key is "return" and the associated value is a list of the note properties as listed above in the discussion of the returned note dictionaries, e.g. ["note_id", "pitch", "velocity"]. The effect of this will be that the returned note dictionaries will only contain the key-value pairs for the specified properties, which can be useful to improve patch performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0.

get_notes_extended

Parameters:
`from_pitch` [int]
`pitch_span` [int]
`from_time` [float]
`time_span` [float]

`from_time` and `time_span` are given in beats.

Returns a dictionary of notes that have their start times in the given area, as a list of note dictionaries. Each note dictionary consists of the following key-value pairs:
`note_id` : [int] the unique note identifier.
`pitch` : [int] the MIDI note number, 0...127, 60 is C3.
`start_time` : [float] the note start time in beats of absolute clip time.
`duration` : [float] the note length in beats.
`velocity` : [float] the note velocity, 0 ... 127.

`mute` : [bool] 1 = the note is deactivated.
`probability` : [float] the chance that the note will be played:
1.0 = the note is always played;
0.0 = the note is never played.
`velocity_deviation` : [float] the range of velocity values at which the note can be played:
0.0 = no deviation; the note will always play at the velocity specified by the *velocity* property
-127.0 to 127.0 = the note will be assigned a velocity value between *velocity* and *velocity + velocity_deviation*, inclusive; if the resulting range exceeds the limits of MIDI velocity (0 to 127), then it will be clamped within those limits.
`release_velocity` : [float] the note release velocity.

It is possible to optionally provide the arguments to this function in the form of a single dictionary instead. The dictionary must include all of the parameter names given above as its keys; the associated values are the parameter values you wish to pass to the function.

If you use this method, you can optionally provide an additional key-value pair: the key is "return" and the associated value is a list of the note properties as listed above in the discussion of the returned note dictionaries, e.g. ["note_id", "pitch", "velocity"]. The effect of this will be that the returned note dictionaries will only contain the key-value pairs for the specified properties, which can be useful to improve patch performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0. Replaces get_notes.

get_selected_notes_extended

Parameter:
`dict (optional)` [dict]
(See below for a discussion of this argument).

Returns a dictionary of the selected notes in the clip, as a list of note dictionaries. Each note dictionary consists of the following key-value pairs:
`note_id` : [int] the unique note identifier.
`pitch` : [int] the MIDI note number, 0...127, 60 is C3.
`start_time` : [float] the note start time in beats of absolute clip time.
`duration` : [float] the note length in beats.
`velocity` : [float] the note velocity, 0 ... 127.

`mute` : [bool] 1 = the note is deactivated.
`probability` : [float] the chance that the note will be played:
1.0 = the note is always played;
0.0 = the note is never played.
`velocity_deviation` : [float] the range of velocity values at which the note can be played:
0.0 = no deviation; the note will always play at the velocity specified by the *velocity* property
-127.0 to 127.0 = the note will be assigned a velocity value between *velocity* and *velocity + velocity_deviation*, inclusive; if the resulting range exceeds the limits of MIDI velocity (0 to 127), then it will be clamped within those limits.
`release_velocity` : [float] the note release velocity.

It is possible to optionally provide a single [dict] argument to this function, containing a single key-value pair: the key is "return" and the associated value is a list of the note properties as listed above in the discussion of the returned note dictionaries, e.g. ["note_id", "pitch", "velocity"]. The effect of this will be that the returned note dictionaries will only contain the key-value pairs for the specified properties, which can be useful to improve patch performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0. Replaces get_selected_notes.

move_playing_pos

Parameter: `beats`
`beats` [float] relative jump distance in beats. Negative beats jump backwards.
Jumps by given amount, unquantized.
Unwarped audio clips, recording audio clips and recording non-overdub MIDI clips cannot jump.

move_warp_marker

Parameters: `beat_time` [float]
`beat_time_distance` [float]
Moves the warp marker specified by *beat_time* the specified beat time distance.

quantize

Parameter:
`quantization_grid` [int]
`amount` [float]
Quantizes all notes in the clip to the quantization_grid taking the song's swing_amount into account.

quantize_pitch

Parameter:
`pitch` [int]
`quantization_grid` [int]
`amount` [float]
Same as *quantize*, but only for notes in the given pitch.

remove_notes_by_id

Parameter:
`list` of note IDs.
Deletes all notes associated with the provided IDs.
Provided note IDs must be associated with existing notes in the clip. Existing notes can be queried with `get_notes_extended`.

Available since Live 11.0.

remove_notes_extended

Parameter:
`from_pitch` [int]
`pitch_span` [int]
`from_time` [float]
`time_span` [float]
Deletes all notes that start in the given area. `from_time` and `time_span` are given in beats.

Available since Live 11.0. Replaces remove_notes.

remove_warp_marker

Parameter: `beat_time` [float]
Removes the warp marker at the given beat time.

scrub

Parameter: `beat_time` [float]
Scrub the clip to a time, specified in beats. This behaves exactly like scrubbing with the mouse; the scrub will respect Global Quantization, starting and looping in time with the transport. The scrub will continue until stop_scrub() is called.

select_all_notes

Use this function to process all notes of a clip, independent of the current selection.

Output:
`select_all_notes id 0`

For MIDI clips only.

select_notes_by_id

Parameter:
`list` of note IDs.
Selects all notes associated with the provided IDs.

Note that this function will *not* print a warning or error if the list contains nonexistent IDs.

Available since Live 11.0.6

set_fire_button_state

Parameter: `state` [bool]
If the state is set to 1, Live simulates pressing the clip start button until the state is set to 0, or until the clip is otherwise stopped.

stop

Same effect as pressing the stop button of the track, but only if this clip is actually playing or recording. If this clip is triggered or if another clip in this track is playing, it has no effect.

stop_scrub

Stops an active scrub on a clip.

ClipSlot

Canonical Path 51
Children 52
clip 52
Properties 52
color 52
color_index 52
controls_other_clips 52
has_clip 53
has_stop_button 53
is_group_slot 53
is_playing 53
is_recording 53
is_triggered 54
playing_status 54
will_record_on_start 54
Functions 54
create_audio_clip 54
create_clip 55
delete_clip 55
duplicate_clip_to 55
fire 55
set_fire_button_state 55
stop 55

This class represents an entry in Live's Session View matrix.

The properties `playing_status`, `is_playing` and `is_recording` are useful for clip slots of Group Tracks. These are always empty and represent the state of the clips in the tracks within the Group Track.

Canonical Path

```
live_set tracks N clip_slots M
```

Children

clip Clip

Access: get

`id 0` if slot is empty

Properties

color long

Access: get observe

The color of the first clip in the Group Track if the clip slot is a Group Track slot.

color_index long

Access: get observe

The color index of the first clip in the Group Track if the clip slot is a Group Track slot.

controls_other_clips bool

Access: get observe

1 for a Group Track slot that has non-deactivated clips in the tracks within its group. Control of empty clip slots doesn't count.

has_clip bool

Access: get observe

1 = a clip exists in this clip slot.

has_stop_button bool

Access: get set observe

1 = this clip stops its track (or tracks within a Group Track).

is_group_slot bool

Access: get

1 = this clip slot is a Group Track slot.

is_playing bool

Access: get

1 = playing_status != 0, otherwise 0.

is_recording bool

Access: get

1 = playing_status == 2, otherwise 0.

is_triggered bool

Access: `get` `observe`

1 = clip slot button (Clip Launch, Clip Stop or Clip Record) or button of contained clip are blinking.

playing_status int

Access: `get` `observe`

0 = all clips in tracks within a Group Track stopped or all tracks within a Group Track are empty.
1 = at least one clip in a track within a Group Track is playing.
2 = at least one clip in a track within a Group Track is playing or recording.
Equals 0 if this is not a clip slot of a Group Track.

will_record_on_start bool

Access: `get`

1 = clip slot will record on start.

Functions

create_audio_clip

Parameter: `path`
Given an absolute path to a valid audio file in a supported format, creates an audio clip that references the file in the clip slot. Throws an error if the clip slot doesn't belong to an audio track or if the track is frozen.

create_clip

Parameter: `length`
Length is given in beats and must be a greater value than 0.0. Can only be called on empty clip slots in MIDI tracks.

delete_clip

Deletes the contained clip.

duplicate_clip_to

Parameter: `target_clip_slot` [ClipSlot]
Duplicates the slot's clip to the given clip slot, overriding the target clip slot's clip if it's not empty.

fire

Parameter: `record_length (optional)`
`launch_quantization (optional)`
Fires the clip or triggers the Stop Button, if any. Starts recording if slot is empty and track is armed. Starts recording of armed and empty tracks within a Group Track if Preferences->Launch->Start Recording on Scene Launch is ON. If *record_length* is provided, the slot will record for the given length in beats. *launch_quantization* overrides the global quantization if provided.

set_fire_button_state

Parameter: `state` [bool]
1 = Live simulates pressing of Clip Launch button until the state is set to 0 or until the slot is stopped otherwise.

stop

Stops playing or recording clips in this track or the tracks within the group, if any. It doesn't matter on which slot of the track you call this function.

CompressorDevice

Properties 57
available_input_routing_channels 57
available_input_routing_types 57
input_routing_channel 58
input_routing_type 58

This class represents a Compressor device in Live.
A CompressorDevice shares all of the children, functions and properties of a Device; listed below are the members unique to it.

Properties

available_input_routing_channels dict

Access: get observe

The list of available source channels for the compressor's input routing in the sidechain. It's represented as a dictionary with the following key:
`available_input_routing_channels` [list]
The list contains dictionaries as described in *input_routing_channel*.

available_input_routing_types dict

Access: get observe

The list of available source types for the compressor's input routing in the sidechain. It's represented as a dictionary with the following key:

`available_input_routing_types` [list]
The list contains dictionaries as described in *input_routing_type*.

input_routing_channel dict

Access: get set observe

The currently selected source channel for the compressor's input routing in the sidechain. It's represented as a dictionary with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the compressor's *available_input_routing_channels*.

input_routing_type dict

Access: get set observe

The currently selected source type for the compressor's input routing in the sidechain. It's represented as a dictionary with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the track's *available_input_routing_types*.

ControlSurface

Canonical Path 59
Properties 60
pad_layout 60
Functions 60
get_control 61
get_control_names 61
grab_control 61
grab_midi 61
register_midi_control 61
release_control 62
release_midi 62
send_midi 62
send_receive_sysex 62

A ControlSurface can be reached either directly by the root path `control_surfaces N` or by getting a list of active control surface IDs, via calling *get control_surfaces* on an Application object. The latter list is in the same order in which control surfaces appear in Live's Link/MIDI Preferences. Note the same order is not guaranteed when getting a control surface via the `control_surfaces N` path.

A control surface can be thought of as a software layer between the Live API and, in this case, Max for Live. Individiual controls on the surface are represented by objects that can be grabbed and released via Max for Live, to obtain and give back exclusive control (see *grab_control* and *release_control*). In this way, parts of the hardware can be controlled via Max for Live while other parts can retain their default functionality.

Additionally, Live offers a special `MaxForLive` control surface that has a *register_midi_control* function. Using this, Max for Live developers can set up entirely custom control surfaces by adding and grabbing arbitrary controls.

Canonical Path

```
control_surfaces N
```

Properties

pad_layout symbol

Access: get observe

The active pad layout.

On Push 2 and 3, the layout can be changed with the Note and Session buttons and depends on the loaded instrument. Layout variants can be selected by pressing the Layout button.

Available layouts are:\

- Melodic mode - the device chain is empty or an Instrument is loaded
  `note.melodic.64_notes` - Melodic: 64 Notes
  `note.melodic.64_notes_and_macro_variations` - Melodic: 64 Notes + Macro Variations
  `note.melodic.sequencer` - Melodic: Sequencer
  `note.melodic.sequencer_and_32_notes` - Melodic: Sequencer + 32 Notes
- Drums mode - a Drum Rack is loaded
  `note.drums.macro_variations` - Drums: Macro Variations
  `note.drums.64_pads` - Drums: 64 Pads
  `note.drums.loop_selector` - Drums: Loop Selector
  `note.drums.16_velocities` - Drums: 16 Velocities
  `note.drums.16_pitches` - Drums: 16 Pitches
- Session mode - the Session button was pressed
  `session` - Session is active

Functions

get_control

Parameter: `name`
Returns the control with the given name.

get_control_names

Returns the list of all control names.

grab_control

Parameter: `control`
Take ownership of the *control*. This releases all standard functionality of the control, so that it can be used exclusively via Max for Live.

grab_midi

Forward MIDI messages received by the control surface script from the control surface to Max for Live.

Note: the control surface script will only receive those channel messages from Live's engine that it explicitly requests. For example, a script might use a specific note message to toggle transport in Live; it will thus request that this note message be forwarded to it from Live.
Messages used for purely real-time purposes, on the other hand, will often bypass the script and instead just be sent to Live's tracks; this is true, for example, of Push's pads in Note (but not Session) mode. Accordingly, the API object will not output these real-time pad messages; to work with track messages, use objects such as `midiin`.

register_midi_control

Parameters:
`name` [symbol]
`status` [int]

`number` [int]

(*MaxForLive* control surface only) Register a MIDI control defined by *status* and *number*. Supported status codes are *144* (note on), *176* (continuous control) and *224* (pitchbend).
Returns the LOM ID associated with the control.
Once a control is registered and grabbed via *grab_control*, Live will forward associated MIDI messages that it receives to Max for Live. Max for Live can send values to the control (e.g. to light an LED) by calling *send_value* on the control object.

release_control

Parameter: `control`
Re-establishes the standard functionality for the control.

release_midi

Stop forwarding MIDI messages received from the control surface to Max for Live.

send_midi

Parameter: `midi_message` [list of int]
Send *midi_message* to the control surface.

send_receive_sysex

Parameters:
`sysex_message` [list of int]
`timeout` [symbol, int]
Send *sysex_message* to the control surface and await a response.
If the message is followed by the word *timeout* and a float, this sets the response timeout accordingly. The default timeout value is 0.2.
If the response times out and MIDI has not been grabbed via *grab_midi*, it's not forwarded to Max for Live. If MIDI has been grabbed via Max for Live, received messages are always forwarded, but the timeout is still reported.

CuePoint

Canonical Path 63
Properties 63
  name 63
  time 63
Functions 64
  jump 64

Represents a locator in the Arrangement View.

Canonical Path

```
live_set cue_points N
```

Properties

name symbol

Access: get set observe

time float

Access: get observe

Arrangement position of the marker in beats.

Functions

jump

Set current Arrangement playback position to marker, quantized if song is playing.

Device.View

Canonical Paths 65
Properties 65
is_collapsed 65

Representing the view aspects of a Device.

Canonical Paths

```
live_set tracks N devices M view
```

```
live_set tracks N devices M chains L devices K view
```

```
live_set tracks N devices M return_chains L devices K view
```

Properties

is_collapsed bool

Access: get set observe

1 = the device is shown collapsed in the device chain.

Device

Canonical Paths 67
Children 68
  parameters 68
  view 68
Properties 68
  can_have_chains 68
  can_have_drum_pads 69
  class_display_name 69
  class_name 69
  is_active 69
  name 69
  type 70
  latency_in_samples 70
  latency_in_ms 70
  can_compare_ab 70
  is_using_compare_preset_b 70
Functions 71
  store_chosen_bank 71
  save_preset_to_compare_ab_slot 71

This class represents a MIDI or audio device in Live.

Canonical Paths

```
live_set tracks N devices M
```

```
live_set tracks N devices M chains L devices K
```

```
live_set tracks N devices M return_chains L devices K
```

Children

parameters list of DeviceParameter

Access: get observe

Only automatable parameters are accessible. See DeviceParameter to learn how to modify them.

view Device.View

Access: get

Properties

can_have_chains bool

Access: get

0 for a single device
1 for a device Rack

can_have_drum_pads bool

Access: get

1 for Drum Racks

class_display_name symbol

Access: get

Get the original name of the device (e.g. `Operator`, `Auto Filter`).

class_name symbol

Access: get

Live device type such as `MidiChord`, `Operator`, `Limiter`, `MxDeviceAudioEffect`, or `PluginDevice`.

is_active bool

Access: get observe

0 = either the device itself or its enclosing Rack device is off.

name symbol

Access: get set observe

This is the string shown in the title bar of the device.

type int

Access: get

The type of the device. Possible types are: 0 = undefined, 1 = instrument, 2 = audio_effect, 4 = midi_effect.

latency_in_samples int

Access: get observe

Device latency in samples.

latency_in_ms float

Access: get observe

Device latency in milliseconds.

can_compare_ab bool

Access: get

1 for devices that support the AB Compare feature. 0 otherwise.

Available since Live 12.3.

is_using_compare_preset_b bool

Access: get set observe

1 if the device has compare preset B loaded. 0 otherwise.
(Only relevant if *can_compare_ab*, otherwise errors.)

Available since Live 12.3.

Functions

store_chosen_bank

Parameters:
`script_index` [int]
`bank_index` [int]
(This is related to hardware control surfaces and is usually not relevant.)

save_preset_to_compare_ab_slot

Save the device state to the other compare AB slot.
(Only relevant if *can_compare_ab*, otherwise errors.)

Available since Live 12.3.

DeviceIO

Properties 72
available_routing_channels 72
available_routing_types 72
default_external_routing_channel_is_none 73
routing_channel 73
routing_type 73

This class represents an input or output bus of a Live device.

Properties

available_routing_channels dictionary

Access: get observe

The available channels for this input/output bus. The channels are represented as a *dictionary* with the following key:
`available_routing_channels` [list]
The list contains *dictionaries* as described in *routing_channel*.

available_routing_types dictionary

Access: get observe

The available types for this input/output bus. The types are represented as a *dictionary* with the following key:

`available_routing_types` [list]
The list contains *dictionaries* as described in *routing_type*.

default_external_routing_channel_is_none bool

Access: get set

1 = the default routing channel for External routing types is none.

Available since Live 11.0.

routing_channel dictionary

Access: get set observe

The current routing channel for this input/output bus. It is represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to any of the values found in *available_routing_channels*.

routing_type dictionary

Access: get set observe

The current routing type for this input/output bus. It is represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to any of the values found in *available_routing_types*.

DeviceParameter

Canonical Path 74
Properties 75
automation_state 75
default_value 75
is_enabled 75
is_quantized 75
max 76
min 76
name 76
original_name 76
state 76
value 77
display_value 77
value_items 77
Functions 77
re_enable_automation 77
str_for_value 78
__str__ 78

This class represents an (automatable) parameter within a MIDI or audio device. To modify a device parameter, set its `value` property or send its object ID to live.remote~.

Canonical Path

```
live_set tracks N devices M parameters L
```

Properties

automation_state int

Access: get observe

Get the automation state of the parameter.
0 = no automation.
1 = automation active.
2 = automation overridden.

default_value float

Access: get

Get the default value for this parameter.
Only available for parameters that aren't quantized (see *is_quantized*).

is_enabled bool

Access: get

1 = the parameter value can be modified directly by the user, by sending `set` to a live.object, by automation or by an assigned MIDI message or keystroke.
Parameters can be disabled because they are macro-controlled, or they are controlled by a live-remote~ object, or because Live thinks that they should not be moved.

is_quantized bool

Access: get

1 for booleans and enums
0 for int/float parameters
Although parameters like MidiPitch.Pitch appear quantized to the user, they actually have an is_quantized value of 0.

max float

Access: get

Largest allowed value.

min float

Access: get

Lowest allowed value.

name symbol

Access: get

The short parameter name as shown in the (closed) automation chooser.

original_name symbol

Access: get

The name of a Macro parameter before its assignment.

state int

Access: get observe

The active state of the parameter.
0 = the parameter is active and can be changed.
1 = the parameter can be changed but isn't active, so changes won't have an audible effect.
2 = the parameter cannot be changed.

value float

Access: get set observe

The internal value between min and max. Use display_value for the value as visible in the GUI.

display_value float

Access: get set observe

The value as visible in the GUI.

value_items StringVector

Access: get

Get a list of the possible values for this parameter.
Only available for parameters that are quantized (see *is_quantized*).

Functions

re_enable_automation

Re-enable automation for this parameter.

str_for_value

Parameter: `value` [float] Returns: [symbol] String representation of the specified value.

__str__

Returns: [symbol] String representation of the current parameter value.

DriftDevice

Properties 80
- mod_matrix_filter_source_1_index 80
- mod_matrix_filter_source_1_list 80
- mod_matrix_filter_source_2_index 80
- mod_matrix_filter_source_2_list 80
- mod_matrix_lfo_source_index 80
- mod_matrix_lfo_source_list 81
- mod_matrix_pitch_source_1_index 81
- mod_matrix_pitch_source_1_list 81
- mod_matrix_pitch_source_2_index 81
- mod_matrix_pitch_source_2_list 81
- mod_matrix_shape_source_index 82
- mod_matrix_shape_source_list 82
- mod_matrix_source_1_index 82
- mod_matrix_source_1_list 82
- mod_matrix_source_2_index 82
- mod_matrix_source_2_list 83
- mod_matrix_source_3_index 83
- mod_matrix_source_3_list 83
- mod_matrix_target_1_index 83
- mod_matrix_target_1_list 83
- mod_matrix_target_2_index 84
- mod_matrix_target_2_list 84
- mod_matrix_target_3_index 84
- mod_matrix_target_3_list 84
- pitch_bend_range 84
- voice_count_index 85
- voice_count_list 85
- voice_mode_index 85
- voice_mode_list 85

This class represents an instance of a Drift device in Live.
A DriftDevice has all the properties, functions and children of a Device.

Properties

mod_matrix_filter_source_1_index int

Access: get set observe

The index of the available sources for modulating the Filter Frequency for the first modulation slot.

mod_matrix_filter_source_1_list StringVector

Access: get

The list of the available sources for modulating the Filter Frequency for the first modulation slot.

mod_matrix_filter_source_2_index int

Access: get set observe

The index of the available sources for modulating the Filter Frequency for the second modulation slot.

mod_matrix_filter_source_2_list StringVector

Access: get

The list of the available sources for modulating the Filter Frequency for the second modulation slot.

mod_matrix_lfo_source_index int

Access: get set observe

The index of the available sources for modulating the LFO Amount.

mod_matrix_lfo_source_list StringVector

Access: get

The list of the available sources for modulating the LFO Amount.

mod_matrix_pitch_source_1_index int

Access: get set observe

The index of the available sources for modulating the Pitch for the first modulation slot.

mod_matrix_pitch_source_1_list StringVector

Access: get

The list of the available sources for modulating the Pitch for the first modulation slot.

mod_matrix_pitch_source_2_index int

Access: get set observe

The index of the available sources for modulating the Pitch for the second modulation slot.

mod_matrix_pitch_source_2_list StringVector

Access: get

The list of the available sources for modulating the Pitch for the second modulation slot.

mod_matrix_shape_source_index int

Access: get set observe

The index of the available sources for modulating Shape.

mod_matrix_shape_source_list StringVector

Access: get

The list of the available sources for modulating Shape.

mod_matrix_source_1_index int

Access: get set observe

The index of the available sources for the first custom modulation slot.

mod_matrix_source_1_list StringVector

Access: get

The list of the available sources for the first custom modulation slot.

mod_matrix_source_2_index int

Access: get set observe

The index of the available sources for the second custom modulation slot.

mod_matrix_source_2_list StringVector

Access: get

The list of the available sources for the second custom modulation slot.

mod_matrix_source_3_index int

Access: get set observe

The index of the available sources for the third custom modulation slot.

mod_matrix_source_3_list StringVector

Access: get

The list of the available sources for the third custom modulation slot.

mod_matrix_target_1_index int

Access: get set observe

The index of the available targets for the first custom modulation slot.

mod_matrix_target_1_list StringVector

Access: get

The list of the available targets for the first custom modulation slot.

mod_matrix_target_2_index int

Access: get set observe

The index of the available targets for the second custom modulation slot.

mod_matrix_target_2_list StringVector

Access: get

The list of the available targets for the second custom modulation slot.

mod_matrix_target_3_index int

Access: get set observe

The index of the available targets for the third custom modulation slot.

mod_matrix_target_3_list StringVector

Access: get

The list of the available targets for the third custom modulation slot.

pitch_bend_range int

Access: get set observe

The amount for the MIDI Pitch Bend range in semitones.

voice_count_index int

Access: get set observe

The index of the voice count parameter.

voice_count_list StringVector

Access: get

The list of available voice count settings.

voice_mode_index int

Access: get set observe

The index of the voice mode utilized by Drift.

voice_mode_list StringVector

Access: get

The list of available voice modes.

DrumCellDevice

Properties 86
gain 86

This class represents an instance of a Drum Sampler device in Live.
A DrumCell has all the properties, functions and children of a Device. Listed below are members unique to DrumCell Device.

Properties

gain float

Access: get set observe

The sample gain, as normalized value.

DrumChain

Properties 87
in_note 87
out_note 87
choke_group 88

This class represents a Drum Rack device chain in Live.

A DrumChain is a type of Chain, meaning that it has all the children, properties and functions that a Chain has. Listed below are the members unique to DrumChain.

Properties

in_note int

Access: get set observe

Get/set the MIDI note that will trigger this chain. The value -1 corresponds to the "All Notes" setting in the UI.

Available since Live 12.3

out_note int

Access: get set observe

Get/set the MIDI note sent to the devices in the chain.

choke_group int

Access: get set observe

Get/set the chain's choke group.

DrumPad

Canonical Path 89
Children 89
  chains 89
Properties 90
  mute 90
  name 90
  note 90
  solo 90
Functions 90
  delete_all_chains 90

This class represents a Drum Rack pad in Live.

Canonical Path

```
live_set tracks N devices M drum_pads L
```

Children

chains Chain

Access: get observe

Properties

mute bool

Access: get set observe

1 = muted

name symbol

Access: get observe

note int

Access: get

solo bool

Access: get set observe

1 = soloed (Solo switch on)
Does not automatically turn Solo off in other chains.

Functions

delete_all_chains

Eq8Device.View

Properties 91
selected_band 91

Represents the view aspects of an Eq8Device.
An Eq8Device.View has all the children, properties and functions of a Device.View. Listed below are members unique to it.

Properties

selected_band int

Access: get set observe

The index of the currently selected filter band.

Eq8Device

Properties 92
edit_mode 92
global_mode 92
oversample 93

This class represents an instance of an EQ Eight device in Live.
An Eq8Device has all the properties, functions and children of a Device. Listed below are members unique to Eq8Device.

Properties

edit_mode bool

Access: get set observe

Access to EQ Eight's edit mode, which toggles the channel currently available for editing. The available edit modes depend on the global mode (see `global_mode`) and are encoded as follows:

In L/R mode: 0 = L, 1 = R
In M/S mode: 0 = M, 1 = S
In Stereo mode: 0 = A, 1 = B (inactive)

global_mode int

Access: get set observe

Access to EQ Eight's global mode. The modes are encoded as follows:

0 = Stereo
1 = L/R
2 = M/S

oversample bool

Access: get set observe

Access to EQ Eight's Oversampling parameter. 0 = Off, 1 = On.

Groove

Canonical Paths 94
Children 94
- base 95
- name 95
- quantization_amount 95
- random_amount 95
- timing_amount 95
- velocity_amount 96

This class represents a groove in Live.

Available since Live 11.0.
All grooves are stored in Live's groove pool.

Canonical Paths

```
live_set groove_pool grooves N
```

```
live_set tracks N clip_slots M clip groove
```

Children

base int

Access: get set

Get/set the groove's base grid (index based setter).
0 = 1/4
1 = 1/8
2 = 1/8T
3 = 1/16
4 = 1/16T
5 = 1/32

name symbol

Access: get set observe

Get/set/observe the name of the groove.

quantization_amount float

Access: get set observe

Get/set/observe the groove's quantization amount.

random_amount float

Access: get set observe

Get/set/observe the groove's random amount.

timing_amount float

Access: get set observe

Get/set/observe the groove's timing amount.

velocity_amount float

Access: get set observe

Get/set/observe the groove's velocity amount.

GroovePool

Canonical Path 97
Children 97
grooves 97

This class represents the groove pool in Live. It provides access to the current set's list of grooves.

Canonical Path

```
live_set groove_pool
```

Children

grooves list of Groove

Access: `get` `observe`

List of grooves in the groove pool from top to bottom, can be accessed via index.

HybridReverbDevice

Properties 98
ir_attack_time 98
ir_category_index 98
ir_category_list 99
ir_decay_time 99
ir_file_index 99
ir_file_list 99
ir_size_factor 99
ir_time_shaping_on 100

This class represents an instance of a Hybrid Reverb device in Live.
A HybridReverbDevice has all the properties, functions and children of a Device. Listed below are members unique to HybridReverbDevice.

Properties

ir_attack_time float

Access: get set observe

The attack time of the amplitude envelope for the impulse response, in seconds.

ir_category_index int

Access: get set observe

The index of the selected impulse response category.

ir_category_list StringVector

Access: get

The list of impulse response categories.

ir_decay_time float

Access: get set observe

The decay time of the amplitude envelope for the impulse response, in seconds.

ir_file_index int

Access: get set observe

The index of the selected impulse response files from the current category.

ir_file_list StringVector

Access: get observe

The list of impulse response files from the selected category.

ir_size_factor float

Access: get set observe

The relative size of the impulse response, 0.0 to 1.0.

ir_time_shaping_on bool

Access: get set observe

Enables transforming the current selected impulse response with an amplitude envelope and size parameter.
1 = enabled.

LooperDevice

Properties 101
loop_length 101
overdub_after_record 102
record_length_index 102
record_length_list 102
tempo 102
Functions 102
clear 103
double_speed 103
half_speed 103
double_length 103
half_length 103
record 103
overdub 103
play 103
stop 104
undo 104
export_to_clip_slot 104

This class represents an instance of a Looper device in Live.
An LooperDevice has all the properties, functions and children of a Device. Listed below are members unique to LooperDevice.

Properties

loop_length float

Access: get observe

The length of Looper's buffer.

overdub_after_record bool

Access: get set observe

1 = Looper will switch to overdub after recording, when recording a fixed number of bars. 0 = switch to playback without overdubbing.

record_length_index int

Access: get set observe

Access to the Record Length chooser entry index.

record_length_list StringVector

Access: get

Access to the list of Record Length chooser entry strings.

tempo float

Access: get observe

The tempo of Looper's buffer.

Functions

clear

Erase Looper's recorded content.

double_speed

Double the speed of Looper's playback.

half_speed

Halve the speed of Looper's playback.

double_length

Double the length of Looper's buffer.

half_length

Halve the length of Looper's buffer.

record

Record incoming audio.

overdub

Play back while adding additional layers of incoming audio.

play

Play back without overdubbing.

stop

Stop Looper's playback.

undo

Erase everything that was recorded since the last time Overdub was enabled. Calling a second time will restore the material erased by the previous undo operation.

export_to_clip_slot

Parameter: `clip_slot` [ClipSlot]
The target clip slot.

Given a valid LOM ID of an empty clip slot on a non-frozen audio track, will export Looper's content to a clip in that slot. This is similar to using the Drag Me! control on the Looper device, and the same restrictions apply: the audio engine must be turned on, the Looper must actually hold audio content, the content must have a fixed length (i.e. Looper must not be recording), etc.

MaxDevice

Properties 105
audio_inputs 105
audio_outputs 105
midi_inputs 106
midi_outputs 106
Functions 106
get_bank_count 106
get_bank_name 106
get_bank_parameters 106

This class represents a Max for Live device in Live.

A MaxDevice is a type of Device, meaning that it has all the children, properties and functions that a Device has. Listed below are the members unique to MaxDevice.

Properties

audio_inputs list of DeviceIO

Access: get observe

List of the audio inputs that the MaxDevice offers.

audio_outputs list of DeviceIO

Access: get observe

List of the audio outputs that the MaxDevice offers.

midi_inputs list of DeviceIO

Access: get observe

List of the midi inputs that the MaxDevice offers.

Available since Live 11.0.

midi_outputs list of DeviceIO

Access: get observe

List of the midi outputs that the MaxDevice offers.

Available since Live 11.0.

Functions

get_bank_count

Returns: [int] the number of parameter banks.

get_bank_name

Parameters: `bank_index` [int]
Returns: [list of symbols] The name of the parameter bank specified by bank_index.

get_bank_parameters

Parameters: `bank_index` [int]
Returns: [list of ints] The indices of the parameters contained in the bank specified by bank_index. Empty slots are marked as -1. Bank index -1 refers to the "Best of" bank.

MeldDevice

Properties 108
selected_engine 108
unison_voices 108
mono_poly 109
poly_voices 109

This class represents an instance of a Meld device in Live.
A MeldDevice has all the properties, functions and children of a Device.

Properties

selected_engine int

Access: get set observe

Meld's oscillator engine selector. The modes are encoded as follows:
0 = Engine A
1 = Engine B

unison_voices int

Access: get set observe

Selects the Unison voice count. The modes are encoded as follows:

0 = off
1 = two

2 = three
3 = four

mono_poly int

Access: get set observe

Selects the polyphony mode. The modes are encoded as follows:

0 = mono
1 = poly

poly_voices int

Access: get set observe

Selects the polyphony voice count. The modes are encoded as follows:

0 = two
1 = three
2 = four
3 = five
4 = six
5 = eight
6 = twelve

MixerDevice

Canonical Path 110
Children 110
sends 110
cue_volume 111
crossfader 111
left_split_stereo 111
panning 111
right_split_stereo 111
song_tempo 112
track_activator 112
volume 112
Properties 112
crossfade_assign 112
panning_mode 112

This class represents a mixer device in Live. It provides access to volume, panning and other DeviceParameter objects. See DeviceParameter to learn how to modify them.

Canonical Path

```
live_set tracks N mixer_device
```

Children

sends list of DeviceParameter

Access: get observe

One send per return track.

cue_volume DeviceParameter

Access: get

[in master track only]

crossfader DeviceParameter

Access: get

[in master track only]

left_split_stereo DeviceParameter

Access: get

The Track's Left Split Stereo Pan Parameter.

panning DeviceParameter

Access: get

right_split_stereo DeviceParameter

Access: get

The Track's Right Split Stereo Pan Parameter.

song_tempo DeviceParameter

Access: get

[in master track only]

track_activator DeviceParameter

Access: get

volume DeviceParameter

Access: get

Properties

crossfade_assign int

Access: get set observe

0 = A, 1 = none, 2 = B [not in master track]

panning_mode int

Access: get set observe

Access to the Track mixer's pan mode: 0 = Stereo, 1 = Split Stereo.

PluginDevice

Properties 113
is_editor_open 113
presets 113
selected_preset_index 113

This class represents a plug-in device.

A PluginDevice is a type of Device, meaning that it has all the children, properties and functions that a Device has. Listed below are the members unique to PluginDevice.

Properties

is_editor_open bool

Access: get set observe

The opened state of the plug-in's editor window.

presets StringVector

Access: get observe

Get the list of the plug-in's presets.

selected_preset_index int

Access: get set observe

Get/set the index of the currently selected preset.

RackDevice.View

Children 115
selected_drum_pad 115
selected_chain 115
Properties 116
drum_pads_scroll_position 116
is_showing_chain_devices 116

Represents the view aspects of a Rack Device.
A RackDevice.View is a type of Device.View, meaning that it has all the properties that a Device.View has. Listed below are the members unique to RackDevice.View.

Children

selected_drum_pad DrumPad

Access: get set observe

Currently selected Drum Rack pad.
Only available for Drum Racks.

selected_chain Chain

Access: get set observe

Currently selected chain.

Properties

drum_pads_scroll_position int

Access: get set observe

Lowest row of pads visible, range: 0 - 28.
Only available for Drum Racks.

is_showing_chain_devices bool

Access: get set observe

1 = the devices in the currently selected chain are visible.

RackDevice

Children 117
chain_selector 118
chains 118
drum_pads 118
return_chains 118
visible_drum_pads 118
Properties 119
can_show_chains 119
has_drum_pads 119
has_macro_mappings 119
is_showing_chains 119
variation_count 119
selected_variation_index 120
visible_macro_count 120
Functions 120
copy_pad 120
add_macro 120
insert_chain 121
remove_macro 121
randomize_macros 121
store_variation 121
recall_selected_variation 121
recall_last_used_variation 122
delete_selected_variation 122

This class represents a Live Rack Device.
A RackDevice is a type of Device, meaning that it has all the children, properties and functions that a Device has. Listed below are members unique to RackDevice.

Children

chain_selector DeviceParameter

Access: get

Convenience accessor for the Rack's chain selector.

chains list of Chain

Access: get observe

The Rack's chains.

drum_pads list of DrumPad

Access: get observe

All 128 Drum Pads for the topmost Drum Rack. Inner Drum Racks return a list of 0 entries.

return_chains list of Chain

Access: get observe

The Rack's return chains.

visible_drum_pads list of DrumPad

Access: get observe

All 16 visible DrumPads for the topmost Drum Rack. Inner Drum Racks return a list of 0 entries.

Properties

can_show_chains bool

Access: get

1 = The Rack contains an instrument device that is capable of showing its chains in Session View.

has_drum_pads bool

Access: get observe

1 = the device is a Drum Rack with pads. A nested Drum Rack is a Drum Rack without pads.
Only available for Drum Racks.

has_macro_mappings bool

Access: get observe

1 = any of a Rack's Macros are mapped to a parameter.

is_showing_chains bool

Access: get set observe

1 = The Rack contains an instrument device that is showing its chains in Session View.

variation_count int

Access: get observe

The number of currently stored macro variations.

Available since Live 11.0.

selected_variation_index int

Access: get set

Get/set the currently selected variation.

Available since Live 11.0.

visible_macro_count int

Access: get observe

The number of currently visible macros.

Functions

copy_pad

Parameters:
`source_index` [int]
`destination_index` [int]
Copies all content of a Drum Rack pad from a source pad to a destination pad. The source_index and destination_index refer to pad indices inside a Drum Rack.

add_macro

Increases the number of visible macro controls.

Available since Live 11.0.

insert_chain

Parameters: `index` [int] (optional)

Attempts to insert a new chain at the given index, or at the end of the chain list if no index is provided. Throws an error if insertion is not possible.

Available since Live 12.3.

remove_macro

Decreases the number of visible macro controls.

Available since Live 11.0.

randomize_macros

Randomizes the values of eligible macro controls.

Available since Live 11.0.

store_variation

Stores a new variation of the values of all currently mapped macros.

Available since Live 11.0.

recall_selected_variation

Recalls the currently selected macro variation.

Available since Live 11.0.

recall_last_used_variation

Recalls the macro variation that was recalled most recently.

Available since Live 11.0.

delete_selected_variation

Deletes the currently selected macro variation. Does nothing if there is no selected variation.

Available since Live 11.0.

RoarDevice

Properties 123
routing_mode_index 123
routing_mode_list 123
env_listen 123

This class represents an instance of a Roar device in Live.
A RoarDevice has all the properties, functions and children of a Roar Device.

Properties

routing_mode_index int

Access: get set observe

The index of the routing mode utilized by Roar.

routing_mode_list StringVector

Access: get

The list of available routing modes.

env_listen bool

Access: get set observe

Get, set and observe the Envelope Input Listen toogle.

Sample

Canonical Path 126
Properties 126
- beats_granulation_resolution 126
- beats_transient_envelope 126
- beats_transient_loop_mode 126
- complex_pro_envelope 127
- complex_pro_formants 127
- end_marker 127
- file_path 127
- gain 128
- length 128
- sample_rate 128
- slices 128
- slicing_sensitivity 128
- start_marker 129
- texture_flux 129
- texture_grain_size 129
- tones_grain_size 129
- warp_markers 129
- warp_mode 130
- warping 130
- slicing_style 130
- slicing_beat_division 131
- slicing_region_count 131

Functions 131
- gain_display_string 131
- insert_slice 132
- move_slice 132
- remove_slice 132
- clear_slices 132
- reset_slices 132

This class represents a sample file loaded into Simpler.

Canonical Path

```
live_set tracks N devices N sample
```

Properties

beats_granulation_resolution int

Access: get set observe

Get/set which divisions to preserve in the sample in Beats Mode.
0 = 1 Bar
1 = 1/2
2 = 1/4
3 = 1/8
4 = 1/16
5 = 1/32
6 = Transients

beats_transient_envelope float

Access: get set observe

Get/set the duration of a volume fade applied to each segment of audio in Beats Mode.

0 = fastest decay
100 = no fade

beats_transient_loop_mode int

Access: get set observe

Get/set the Transient Loop Mode applied to each segment of audio in Beats Mode.
0 = Off
1 = Loop Forward
2 = Loop Back-and-Forth

complex_pro_envelope float

Access: get set observe

Get/set the Envelope parameter in Complex Pro Mode.

complex_pro_formants float

Access: get set observe

Get/set the Formants parameter in Complex Pro Mode.

end_marker int

Access: get set observe

Get/set the position of the sample's end marker.

file_path unicode

Access: get observe

Get the path of the sample file.

gain float

Access: get set observe

Get/set the sample gain.

length int

Access: get

Get the length of the sample file in sample frames.

sample_rate int

Access: get

The sample rate of the loaded sample.

Available since Live 11.0.

slices list of int

Access: get observe

The positions of all playable slices in the sample, in sample frames. Divide these values by the `sample_rate` to get the slice times in seconds.

Available since Live 11.0.

slicing_sensitivity float

Access: get set observe

Get/set the slicing sensitivity. Values are between 0.0 and 1.0.

start_marker int

Access: get set observe

Get/set the position of the sample's start marker.

texture_flux float

Access: get set observe

Get/set the Flux parameter in Texture Mode.

texture_grain_size float

Access: get set observe

Get/set the Grain Size parameter in Texture Mode.

tones_grain_size float

Access: get set observe

Get/set the Grain Size parameter in Tones Mode.

warp_markers dict/bang

Access: get observe

The Sample's Warp Markers as a dict. Observing this property bangs when the warp_markers change.

The last Warp Marker in the dict is not visible in the Live interface. This hidden, or "shadow" marker is used to calculate the BPM of the last segment.

Available since Live 11.0.

warp_mode int

Access: get set observe

Get/set the Warp Mode.
0 = Beats Mode
1 = Tones Mode
2 = Texture Mode
3 = Re-Pitch Mode
4 = Complex Mode
6 = Complex Pro Mode

warping bool

Access: get set observe

1 = warping is enabled.

slicing_style int

Access: get set observe

Get/set the Slicing Mode.
0 = Transient

1 = Beat
2 = Region
3 = Manual

slicing_beat_division int

Access: get set observe

Get/set the slice beat division in Beat Slicing Mode.
0 = 1/16
1 = 1/16T
2 = 1/8
3 = 1/8T
4 = 1/4
5 = 1/4T
6 = 1/2
7 = 1/2T
8 = 1 Bar
9 = 2 Bars
10 = 4 Bars

slicing_region_count int

Access: get set observe

Get/set the number of slice regions in Region Slicing Mode.

Functions

gain_display_string

Returns: [list of symbols] The sample's gain value as a string, e.g. "0.0 dB".

insert_slice

Parameters: `slice_time` [int]
Insert a new slice at the specified time if there is none.

move_slice

Parameters: `source_time` [int] `destination_time` [int]
Move an existing slice to a specified time.

remove_slice

Parameters: `slice_time` [int]
Remove a slice at the specified time if it exists.

clear_slices

Clear all slices created in Manual Slicing Mode.

reset_slices

Reset all edited slices to their original positions.

Scene

Canonical Path 133
Children 133
clip_slots 134
Properties 134
color 134
color_index 134
is_empty 134
is_triggered 134
name 135
tempo 135
tempo_enabled 135
time_signature_numerator 135
time_signature_denominator 136
time_signature_enabled 136
Functions 136
fire 136
fire_as_selected 136
set_fire_button_state 137

This class represents a series of clip slots in Live's Session View matrix.

Canonical Path

```
live_set scenes N
```

Children

clip_slots list of ClipSlot

Access: get observe

Properties

color int

Access: get set observe

The RGB value of the scene's color in the form `0x00rrggbb` or (2^16 * red) + (2^8) * green + blue, where red, green and blue are values from 0 (dark) to 255 (light).

When setting the RGB value, the nearest color from the Scene color chooser is taken.

color_index long

Access: get set observe

The color index of the scene.

is_empty bool

Access: get

1 = none of the slots in the scene is filled.

is_triggered bool

Access: get observe

1 = scene is blinking.

name symbol

Access: get set observe

The name of the scene.

tempo float

Access: get set observe

The scene's tempo.
Returns -1 if the scene tempo is disabled.

tempo_enabled bool

Access: get set observe

The active state of the scene tempo.
When disabled, the scene will use the song's tempo,
and the tempo value returned will be -1.

time_signature_numerator int

Access: get set observe

The scene's time signature numerator.
Returns -1 if the scene time signature is disabled.

time_signature_denominator int

Access: get set observe

The scene's time signature denominator.
Returns -1 if the scene time signature is disabled.

time_signature_enabled bool

Access: get set observe

The active state of the scene time signature.
When disabled, the scene will use the song's time signature,
and the time signature values returned will be -1.

Functions

fire

Parameter: `force_legato (optional)` [bool]
`can_select_scene_on_launch (optional)` [bool]
Fire all clip slots contained within the scene and select this scene.
Starts recording of armed and empty tracks within a Group Track in this scene if Preferences->Launch->Start Recording on Scene Launch is ON.
Calling with force_legato = 1 (default = 0) will launch all clips immediately in Legato, independent of their launch mode.
When calling with can_select_scene_on_launch = 0 (default = 1) the scene is fired without selecting it.

fire_as_selected

Parameter: `force_legato (optional)` [bool]
Fire the selected scene, then select the next scene.
It doesn't matter on which scene you are calling this function.
Calling with force_legato = 1 (default = 0) will launch all clips immediately in Legato, independent of their launch mode.

set_fire_button_state

Parameter: `state` [bool]
If the state is set to 1, Live simulates pressing of scene button until the state is set to 0 or until the scene is stopped otherwise.

ShifterDevice

Properties 138
pitch_bend_range 138
pitch_mode_index 138

This class represents an instance of the Shifter audio effect.
A ShifterDevice is a type of device, meaning that it has all the children, properties and functions that a device has. Listed below are members unique to ShifterDevice.

Properties

pitch_bend_range int

Access: get set observe

The pitch bend range used in MIDI Pitch Mode.

pitch_mode_index int

Access: get set observe

The current pitch mode index: 0 = Internal, 1 = MIDI

SimplerDevice.View

Properties 139
selected_slice 139

Represents the view aspects of a SimplerDevice.
A SimplerDevice.View is a type of Device.View, meaning that it has all the properties that a Device.View has. Listed below are the members unique to SimplerDevice.View.

Properties

selected_slice int

Access: get set observe

The currenctly selected slice, identified by its slice time.

SimplerDevice

Children 140
sample 140
Properties 141
can_warp_as 141
can_warp_double 141
can_warp_half 141
multi_sample_mode 141
pad_slicing 142
playback_mode 142
playing_position 142
playing_position_enabled 142
retrigger 142
slicing_playback_mode 143
voices 143
Functions 143
crop 143
guess_playback_length 143
reverse 143
warp_as 144
warp_double 144
warp_half 144
replace_sample 144

This class represents an instance of Simpler.
A SimplerDevice is a type of device, meaning that it has all the children, properties and functions that a device has. Listed below are members unique to SimplerDevice.

Children

sample Sample

Access: get observe

The sample currently loaded into Simpler.

Properties

can_warp_as bool

Access: get observe

1 = warp_as is available.

can_warp_double bool

Access: get observe

1 = warp_double is available.

can_warp_half bool

Access: get observe

1 = warp_half is available.

multi_sample_mode bool

Access: get observe

1 = Simpler is in multisample mode.

pad_slicing bool

Access: get set observe

1 = slices can be added in Slicing Mode by playing notes which are not yet assigned to existing slices.

playback_mode int

Access: get set observe

Get/set Simpler's playback mode.
0 = Classic Mode
1 = One-Shot Mode
2 = Slicing Mode

playing_position float

Access: get observe

The current playing position in the sample, expressed as a value between 0. and 1.

playing_position_enabled bool

Access: get observe

1 = Simpler is playing back the sample and showing the playing position.

retrigger bool

Access: get set observe

1 = Retrigger is enabled in Simpler.

slicing_playback_mode int

Access: get set observe

Get/set Simpler's Slicing Playback Mode.
0 = Mono
1 = Poly
2 = Thru

voices int

Access: get set observe

Get/set the number of Voices.

Functions

crop

Crop the loaded sample to the active region between the start and end markers.

guess_playback_length

Returns: [float] An estimated beat time for the playback length between the start and end markers.

reverse

Reverse the loaded sample.

warp_as

Parameters: `beats` [int]
Warp the active region between the start and end markers as the specified number of beats.

warp_double

Double the playback tempo of the active region between the start and end markers.

warp_half

Halve the playback tempo for the active region between the start and end markers.

replace_sample

Parameters: `file_path` [symbol]
Given an absolute path to a valid audio file in a supported format, replaces the loaded sample with the one specified.

Song.View

Canonical Path 145
Children 145
detail_clip 145
highlighted_clip_slot 146
mod_mapping_device 146
mod_mapping_parameter 146
selected_chain 147
selected_parameter 147
selected_scene 147
selected_track 147
Properties 147
draw_mode 147
follow_song 148
Functions 148
select_device 148

This class represents the view aspects of a Live document: the Session and Arrangement Views.

Canonical Path

```
live_set view
```

Children

detail_clip Clip

Access: get set observe

The clip currently displayed in the Live application's Detail View.

highlighted_clip_slot ClipSlot

Access: get set

The slot highlighted in the Session View.

mod_mapping_device Device

Access: get set observe

When assigned a device id, modulation mapping mode is active. When assigned `id 0`, mapping mode is not active. Affects only Push 3 at this time.

The device value that is used to enable modulation mapping is also used for the following purposes:
- When the device is removed, mapping mode is exited automatically without reporting a chosen parameter for mapping.

- When a parameter is chosen that belongs to the assigned device, this parameter will not be chosen for mapping.

mod_mapping_parameter DeviceParameter

Access: get observe

The parameter selected for modulation mapping, or `id 0`.

Observing this property outputs the id of a parameter that is selected in the Live UI or that is chosen for mapping on Push.

Reports `id 0` when no parameter is chosen, for example when a parameter is deselected in the Live UI.

selected_chain Chain

Access: get set observe

The highlighted chain, or "id 0"

selected_parameter DeviceParameter

Access: get observe

The selected parameter, or "id 0"

selected_scene Scene

Access: get set observe

selected_track Track

Access: get set observe

Properties

draw_mode bool

Access: get set observe

Reflects the state of the envelope/automation Draw Mode Switch in the transport bar, as toggled with Cmd/Ctrl-B.
0 = breakpoint editing (shows arrow), 1 = drawing (shows pencil)

follow_song bool

Access: `get` `set` `observe`

Reflects the state of the Follow switch in the transport bar as toggled with Cmd/Ctrl-F.
0 = don't follow playback position, 1 = follow playback position

Functions

select_device

Parameter: `id NN`
Selects the given device object in its track.
You may obtain the id using a live.path or by using `get devices` on a track, for example.
The track containing the device will not be shown automatically, and the device gets the appointed device (blue hand) only if its track is selected.

Song

Canonical Path 151
Children 151
- cue_points 151
- return_tracks 152
- scenes 152
- tracks 152
- visible_tracks 152
- master_track 152
- view 152
- groove_pool 153
- tuning_system 153

Properties 153
- appointed_device 153
- arrangement_overdub 153
- back_to_arranger 154
- can_capture_midi 154
- can_jump_to_next_cue 154
- can_jump_to_prev_cue 154
- can_redo 154
- can_undo 155
- clip_trigger_quantization 155
- count_in_duration 155
- current_song_time 156
- exclusive_arm 156
- exclusive_solo 156
- file_path 156
- groove_amount 156
- is_ableton_link_enabled 157
- is_ableton_link_start_stop_sync_enabled 157
- is_counting_in 157
- is_playing 157
- last_event_time 157
- loop 158
- loop_length 158
- loop_start 158

metronome 158
midi_recording_quantization 158
name 159
nudge_down 159
nudge_up 159
tempo_follower_enabled 159
overdub 160
punch_in 160
punch_out 160
re_enable_automation_enabled 160
record_mode 160
root_note 161
scale_intervals 161
scale_mode 161
scale_name 161
select_on_launch 162
session_automation_record 162
session_record 162
session_record_status 162
signature_denominator 162
signature_numerator 163
song_length 163
start_time 163
swing_amount 163
tempo 163
Functions 164
capture_and_insert_scene 164
capture_midi 164
continue_playing 164
create_audio_track 164
create_midi_track 164
create_return_track 164
create_scene 165
delete_scene 165
delete_track 165
delete_return_track 165
duplicate_scene 165
duplicate_track 165
find_device_position 166
force_link_beat_time 166
get_beats_loop_length 166

get_beats_loop_start 166
get_current_beats_song_time 166
get_current_smpte_song_time 167
is_cue_point_selected 167
jump_by 167
jump_to_next_cue 167
jump_to_prev_cue 167
move_device 168
play_selection 168
re_enable_automation 168
redo 168
scrub_by 168
set_or_delete_cue 168
start_playing 169
stop_all_clips 169
stop_playing 169
tap_tempo 169
trigger_session_record 169
undo 169

This class represents a Live Set. The current Live Set is reachable by the root path `live_set`.

Canonical Path

```
live_set
```

Children

cue_points list of CuePoint

Access: get observe

Cue points are the markers in the Arrangement to which you can jump.

return_tracks list of Track

Access: get observe

scenes list of Scene

Access: get observe

tracks list of Track

Access: get observe

visible_tracks list of Track

Access: get observe

A track is visible if it's not part of a folded group. If a track is scrolled out of view it's still considered visible.

master_track Track

Access: get

view Song.View

Access: get

groove_pool GroovePool

Access: get

Live's groove pool.

Available since Live 11.0.

tuning_system TuningSystem

Access: get observe

Live's currently active tuning system.

Properties

appointed_device Device

Access: get observe

The appointed device is the one used by a control surface unless the control surface itself chooses which device to use. It is marked by a blue hand.

arrangement_overdub bool

Access: get set observe

Get/set the state of the MIDI Arrangement Overdub button.

back_to_arranger bool

Access: get set observe

Get/set/observe the current state of the Back to Arrangement button located in Live's transport bar (1 = highlighted). This button is used to indicate that the current state of the playback differs from what is stored in the Arrangement.

Setting this property to 0 will make Live go back to playing the content of the arrangement.

can_capture_midi bool

Access: get observe

1 = Recently played MIDI material exists that can be captured into a Live Track. See *capture_midi*.

can_jump_to_next_cue bool

Access: get observe

0 = there is no cue point to the right of the current one, or none at all.

can_jump_to_prev_cue bool

Access: get observe

0 = there is no cue point to the left of the current one, or none at all.

can_redo bool

Access: get

1 = there is something in the history to redo.

can_undo bool

Access: get

1 = there is something in the history to undo.

clip_trigger_quantization int

Access: get set observe

Reflects the quantization setting in the transport bar.
0 = None
1 = 8 Bars
2 = 4 Bars
3 = 2 Bars
4 = 1 Bar
5 = 1/2
6 = 1/2T
7 = 1/4
8 = 1/4T
9 = 1/8
10 = 1/8T
11 = 1/16
12 = 1/16T
13 = 1/32

count_in_duration int

Access: get observe

The duration of the Metronome's Count-In setting as an index, mapped as follows:
0 = None

1 = 1 Bar
2 = 2 Bars
3 = 4 Bars

current_song_time float

Access: get set observe

The playing position in the Live Set, in beats.

exclusive_arm bool

Access: get

Current status of the exclusive Arm option set in the Live preferences.

exclusive_solo bool

Access: get

Current status of the exclusive Solo option set in the Live preferences.

file_path symbol

Access: get

The path to the current Live Set, in OS-native format. If the Live Set hasn't been saved, the path is empty.

groove_amount float

Access: get set observe

The groove amount from the current set's groove pool (0. - 1.0).

is_ableton_link_enabled bool

Access: get set observe

Enable/disable Ableton Link. The Link toggle in the Live's transport bar must be visible to enable Link.

is_ableton_link_start_stop_sync_enabled bool

Access: get set observe

Enable/disable Ableton Link Start Stop Sync.

is_counting_in bool

Access: get observe

1 = the Metronome is currently counting in.

is_playing bool

Access: get set observe

Get/set if Live's transport is running.

last_event_time float

Access: get

The beat time of the last event (i.e. automation breakpoint, clip end, cue point, loop end) in the Arrangement.

loop bool

Access: get set observe

Get/set the enabled state of the Arrangement loop.

loop_length float

Access: get set observe

Arrangement loop length in beats.

loop_start float

Access: get set observe

Arrangement loop start in beats.

metronome bool

Access: get set observe

Get/set the enabled state of the metronome.

midi_recording_quantization int

Access: get set observe

Get/set the current Record Quantization value.
0 = None
1 = 1/4
2 = 1/8
3 = 1/8T
4 = 1/8 + 1/8T
5 = 1/16
6 = 1/16T
7 = 1/16 + 1/16T
8 = 1/32

name symbol

Access: get

The name of the current Live Set. If the Live Set hasn't been saved, the name is empty.

nudge_down bool

Access: get set observe

1 = the Tempo Nudge Down button in the transport bar is currently pressed.

nudge_up bool

Access: get set observe

1 = the Tempo Nudge Up button in the transport bar is currently pressed.

tempo_follower_enabled bool

Access: get set observe

1 = the Tempo Follower controls the tempo. The Tempo Follower Toggle must be made visible in the preferences for this property to be effective.

overdub bool

Access: get set observe

1 = MIDI Arrangement Overdub is enabled in the transport.

punch_in bool

Access: get set observe

1 = the Punch-In button is enabled in the transport.

punch_out bool

Access: get set observe

1 = the Punch-Out button is enabled in the transport.

re_enable_automation_enabled bool

Access: get observe

1 = the Re-Enable Automation button is on.

record_mode bool

Access: get set observe

1 = the Arrangement Record button is on.

root_note int

Access: get set observe

The root note of the scale currently selected in Live. The root note can be a number between 0 and 11, where 0 = C and 11 = B.

scale_intervals list

Access: get observe

A list of integers representing the intervals in Live's current scale (see *scale_name* and *scale_mode*). An interval is expressed as the difference between the scale degree at the list index and the first scale degree.

scale_mode bool

Access: get set observe

Access to the Scale Mode setting in Live.

When on, key tracks that belong to the currently selected scale are highlighted in Live's MIDI Note Editor, and pitch-based parameters in MIDI Tools and Devices can be edited in scale degrees rather than semitones.

See also *root_note*, *scale_name*, and *scale_intervals*.

scale_name unicode

Access: get set observe

The name of the scale selected in Live, as displayed in the Current Scale Name chooser.

select_on_launch bool

Access: get

1 = the "Select on Launch" option is set in Live's preferences.

session_automation_record bool

Access: get set observe

The state of the Automation Arm button.

session_record bool

Access: get set observe

The state of the Session Overdub button.

session_record_status int

Access: get observe

Reflects the state of the Session Record button.

signature_denominator int

Access: get set observe

signature_numerator int

Access: get set observe

song_length float

Access: get observe

A little more than `last_event_time`, in beats.

start_time float

Access: get set observe

The position in the Live Set where playing will start, in beats.

swing_amount float

Access: get set observe

Range: 0.0 - 1.0; affects MIDI Recording Quantization and all direct calls to `Clip.quantize`.

tempo float

Access: get set observe

Current tempo of the Live Set in BPM, 20.0 ... 999.0. The tempo may be automated, so it can change depending on the current song time.

Functions

capture_and_insert_scene

Capture the currently playing clips and insert them as a new scene below the selected scene.

capture_midi

Parameter: `destination` [int]
0 = auto, 1 = session, 2 = arrangement
Capture recently played MIDI material from audible tracks into a Live Clip.
If *destinaton* is not set or it is set to *auto*, the Clip is inserted into the view currently visible in the focused Live window. Otherwise, it is inserted into the specified view.

continue_playing

From the current playback position.

create_audio_track

Parameter: `index`
Index determines where the track is added, it is only valid between 0 and len(song.tracks). Using an index of -1 will add the new track at the end of the list.

create_midi_track

Parameter: `index`
Index determines where the track is added, it is only valid between 0 and len(song.tracks). Using an index of -1 will add the new track at the end of the list.

create_return_track

Adds a new return track at the end.

create_scene

Parameter: `index`
Returns: The new scene
Index determines where the scene is added. It is only valid between 0 and len(song.scenes). Using an index of -1 will add the new scene at the end of the list.

delete_scene

Parameter: `index`
Delete the scene at the given index.

delete_track

Parameter: `index`
Delete the track at the given index.

delete_return_track

Parameter: `index`
Delete the return track at the given index.

duplicate_scene

Parameter: `index`
Index determines which scene to duplicate.

duplicate_track

Parameter: `index`
Index determines which track to duplicate.

find_device_position

Parameter:
`device` [live object]
`target` [live object]
`target position` [int]
Returns:
[int] The position in the target's chain where the device can be inserted that is the closest possible to the target position.

force_link_beat_time

Force the Link timeline to jump to Live's current beat time.

get_beats_loop_length

Returns: `bars.beats.sixteenths.ticks` [symbol]
The Arrangement loop length.

get_beats_loop_start

Returns: `bars.beats.sixteenths.ticks` [symbol]
The Arrangement loop start.

get_current_beats_song_time

Returns: `bars.beats.sixteenths.ticks` [symbol]
The current Arrangement playback position.

get_current_smpte_song_time

Parameter: `format`
`format` [int] is the time code type to be returned
0 = the frame position shows the milliseconds
1 = Smpte24
2 = Smpte25
3 = Smpte30
4 = Smpte30Drop
5 = Smpte29
Returns: *hours:min:sec*

[symbol]
The current Arrangement playback position.

is_cue_point_selected

Returns: bool 1 = the current Arrangement playback position is at a cue point

jump_by

Parameter: `beats`
`beats` [float] is the amount to jump relatively to the current position

jump_to_next_cue

Jump to the right, if possible.

jump_to_prev_cue

Jump to the left, if possible.

move_device

Parameter:
`device` [live object]
`target` [live object]
`target position` [int]
Returns: [int] The position in the target's chain where the device was inserted.
Move the device to the specified position in the target chain. If the device cannot be moved to the specified position, the nearest possible position is chosen.

play_selection

Do nothing if no selection is set in Arrangement, or play the current selection.

re_enable_automation

Trigger 'Re-Enable Automation', re-activating automation in all running Session clips.

redo

Causes the Live application to redo the last operation.

scrub_by

Parameter: `beats`
`beats` [float] the amount to scrub relative to the current Arrangement playback position
Same as `jump_by`, at the moment.

set_or_delete_cue

Toggle cue point at current Arrangement playback position.

start_playing

Start playback from the insert marker.

stop_all_clips

Parameter (optional): `quantized`
Calling the function with 0 will stop all clips immediately, independent of the launch quantization. The default is '1'.

stop_playing

Stop the playback.

tap_tempo

Same as pressing the Tap Tempo button in the transport bar. The new tempo is calculated based on the time between subsequent calls of this function.

trigger_session_record

Parameter: `record_length (optional)`
Starts recording in either the selected slot or the next empty slot, if the track is armed. If *record_length* is provided, the slot will record for the given length in beats.
If triggered while recording, recording will stop and clip playback will start.

undo

Causes the Live application to undo the last operation.

SpectralResonatorDevice

Properties 170
frequency_dial_mode 170
midi_gate 170
mod_mode 171
mono_poly 171
pitch_mode 171
pitch_bend_range 171
polyphony 171

This class represents an instance of a Spectral Resonator device in Live.
An SpectralResonatorDevice has all the properties, functions and children of a Device. Listed below are members unique to SpectralResonatorDevice.

Properties

frequency_dial_mode int

Access: get set observe

Get, set and observe the Freq control's mode.
0 = Hertz, 1 = MIDI note values.

midi_gate int

Access: get set observe

Get, set and observe the MIDI gate switch's state.
0 = Off, 1 = On.

mod_mode int

Access: get set observe

Get, set and observe the Modulation Mode.
0 = None, 1 = Chorus, 2 = Wander, 3 = Granular.

mono_poly int

Access: get set observe

Get, set and observe the Mono/Poly switch's state.
0 = Mono, 1 = Poly.

pitch_mode int

Access: get set observe

Get, set and observe the Pitch Mode.
0 = Internal, 1 = MIDI.

pitch_bend_range int

Access: get set observe

Get, set and observe the Pitch Bend Range.\

polyphony int

Access: get set observe

Get, set and observe the Polyphony.
0 = 2, 1 = 4, 2 = 8, 3 = 16 voices.

TakeLane

Canonical Path 173
Children 173
arrangement_clips 173
Properties 174
name 174
Functions 174
create_audio_clip 174
create_midi_clip 174

This class represents a take lane in Live. Tracks in Live can have take lanes in Arrangement View, which are used for comping. If take lanes exist for a track, they can be shown by right-clicking on a track and choosing Show Take Lanes.

Canonical Path

```
live_set tracks N take_lanes M
```

Children

arrangement_clips list of Clip

Access: get observe

The list of this take lane's Arrangement View clip IDs

Properties

name symbol

Access: get set observe

The name as shown in the take lane header.

Functions

create_audio_clip

Parameters:
`file_path` [symbol]
`start_time` [float]
Given a valid audio file in a supported format, passing its absolute path (on Mac, starting with `/Volumes/(drive name)/`) creates an audio clip referencing the file in the arrangement view at the specified `start_time` in beats.

Prints an error if the track is not an audio track, if the track is frozen or if the track is being recorded into. `start_time` must be within the range `[0., 1576800]`.

create_midi_clip

Parameters:
`start_time` [float]
`length` [float]
Creates an empty MIDI clip with the specified `length` in beats and inserts it into the arrangement at the specified `start_time` in beats.

Prints an error if the track is not a MIDI track, if the track is frozen or when the track is currently being recorded into. `start_time` must be within the range `[0., 1576800]`.

this_device

Canonical Path 175

This root path represents the device containing the live.path object to which the `goto this_device` message is sent. The class of this object is `Device`.

Canonical Path

```
live_set tracks N devices M
```

Track.View

Canonical Path 176
Children 176
  selected_device 176
Properties 177
  device_insert_mode 177
  is_collapsed 177
Functions 177
  select_instrument 177

Representing the view aspects of a track.

Canonical Path

```
live_set tracks N view
```

Children

selected_device Device

Access: get observe

The selected device or the first selected device (in case of multi/group selection).

Properties

device_insert_mode int

Access: get set observe

Determines where a device will be inserted when loaded from the browser. 0 = add device at the end, 1 = add device to the left of the selected device, 2 = add device to the right of the selected device.

is_collapsed bool

Access: get set observe

In Arrangement View: 1 = track collapsed, 0 = track opened.

Functions

select_instrument

Returns: bool 0 = there are no devices to select
Selects track's instrument or first device, makes it visible and focuses on it.

Track

Canonical Path 179
Children 180
take_lanes 180
clip_slots 180
arrangement_clips 180
devices 180
group_track 181
mixer_device 181
view 181
Properties 181
arm 181
available_input_routing_channels 181
available_input_routing_types 182
available_output_routing_channels 182
available_output_routing_types 182
back_to_arranger 183
can_be_armed 183
can_be_frozen 183
can_show_chains 183
color 183
color_index 184
fired_slot_index 184
fold_state 184
has_audio_input 184
has_audio_output 185
has_midi_input 185
has_midi_output 185
implicit_arm 185
input_meter_left 185
input_meter_level 186
input_meter_right 186
input_routing_channel 186
input_routing_type 186
is_foldable 187
is_frozen 187

- is_grouped 187
- is_part_of_selection 187
- is_showing_chains 188
- is_visible 188
- mute 188
- muted_via_solo 188
- name 188
- output_meter_left 189
- output_meter_level 189
- output_meter_right 189
- performance_impact 189
- output_routing_channel 189
- output_routing_type 190
- playing_slot_index 190
- solo 190

Functions 191

- create_audio_clip 191
- create_midi_clip 191
- create_take_lane 191
- delete_clip 191
- delete_device 192
- duplicate_clip_slot 192
- duplicate_clip_to_arrangement 192
- insert_device 192
- jump_in_running_session_clip 193
- stop_all_clips 193

This class represents a track in Live. It can either be an audio track, a MIDI track, a return track or the master track. The master track and at least one Audio or MIDI track will be always present. Return tracks are optional.

Not all properties are supported by all types of tracks. The properties are marked accordingly.

Canonical Path

```
live_set tracks N
```

Children

take_lanes list of TakeLane

Access: get observe

The list of this track's take lanes

clip_slots list of ClipSlot

Access: get observe

arrangement_clips list of Clip

Access: get observe

The list of this track's Arrangement View clip IDs

Available since Live 11.0.

devices list of Device

Access: get observe

Includes mixer device.

group_track Track

Access: get

The Group Track, if the Track is grouped. If it is not, *id 0* is returned.

mixer_device MixerDevice

Access: get

view Track.View

Access: get

Properties

arm bool

Access: get set observe

1 = track is armed for recording. [not in return/master tracks]

available_input_routing_channels dictionary

Access: get observe

The list of available source channels for the track's input routing. It's represented as a *dictionary* with the following key:
`available_input_routing_channels` [list]

The list contains *dictionaries* as described in *input_routing_channel*.
Only available on MIDI and audio tracks.

available_input_routing_types dictionary

Access: get observe

The list of available source types for the track's input routing. It's represented as a *dictionary* with the following key:
`available_input_routing_types` [list]
The list contains *dictionaries* as described in *input_routing_type*.
Only available on MIDI and audio tracks.

available_output_routing_channels dictionary

Access: get observe

The list of available target channels for the track's output routing. It's represented as a *dictionary* with the following key:
`available_output_routing_channels` [list]
The list contains *dictionaries* as described in *output_routing_channel*.
Not available on the master track.

available_output_routing_types dictionary

Access: get observe

The list of available target types for the track's output routing. It's represented as a *dictionary* with the following key:
`available_output_routing_types` [list]
The list contains *dictionaries* as described in *output_routing_type*.
Not available on the master track.

back_to_arranger bool

Access: get set observe

Get/set/observe the current state of the Single Track Back to Arrangement button (1 = highlighted). This button is used to indicate that the current state of the playback differs from what is stored in the Arrangement.

Setting this property to 0 will make Live go back to playing the track's arrangement content. For group tracks, this means that all of the tracks that belong to the group and any subgroups will go back to playing the arrangement.

can_be_armed bool

Access: get

0 for return and master tracks.

can_be_frozen bool

Access: get

1 = the track can be frozen, 0 = otherwise.

can_show_chains bool

Access: get

1 = the track contains an Instrument Rack device that can show chains in Session View.

color int

Access: get set observe

The RGB value of the track's color in the form `0x00rrggbb` or (2^16 * red) + (2^8) * green + blue, where red, green and blue are values from 0 (dark) to 255 (light).

When setting the RGB value, the nearest color from the track color chooser is taken.

color_index long

Access: get set observe

The color index of the track.

fired_slot_index int

Access: get observe

Reflects the blinking clip slot.
-1 = no slot fired, -2 = Clip Stop Button fired
First clip slot has index 0.
[not in return/master tracks]

fold_state int

Access: get set

0 = tracks within the Group Track are visible, 1 = Group Track is folded and the tracks within the Group Track are hidden
[only available if `is_foldable` = 1]

has_audio_input bool

Access: get

1 for audio tracks.

has_audio_output bool

Access: get

1 for audio tracks and MIDI tracks with instruments.

has_midi_input bool

Access: get

1 for MIDI tracks.

has_midi_output bool

Access: get

1 for MIDI tracks with no instruments and no audio effects.

implicit_arm bool

Access: get set observe

A second arm state, only used by Push so far.

input_meter_left float

Access: get observe

Smoothed momentary peak value of left channel input meter, 0.0 to 1.0. For tracks with audio output only. This value corresponds to the meters shown in Live. Please take into account that the left/right audio meters put a significant load onto the GUI part of Live.

input_meter_level float

Access: get observe

Hold peak value of input meters of audio and MIDI tracks, 0.0 ... 1.0. For audio tracks it is the maximum of the left and right channels. The hold time is 1 second.

input_meter_right float

Access: get observe

Smoothed momentary peak value of right channel input meter, 0.0 to 1.0. For tracks with audio output only. This value corresponds to the meters shown in Live.

input_routing_channel dictionary

Access: get set observe

The currently selected source channel for the track's input routing. It's represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the track's *available_input_routing_channels*.
Only available on MIDI and audio tracks.

input_routing_type dictionary

Access: get set observe

The currently selected source type for the track's input routing. It's represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the track's *available_input_routing_types*.
Only available on MIDI and audio tracks.

is_foldable bool

Access: get

1 = track can be (un)folded to hide or reveal the contained tracks. This is currently the case for Group Tracks. Instrument and Drum Racks return 0 although they can be opened/closed. This will be fixed in a later release.

is_frozen bool

Access: get observe

1 = the track is currently frozen.

is_grouped bool

Access: get

1 = the track is contained within a Group Track.

is_part_of_selection bool

Access: get

is_showing_chains bool

Access: get set observe

Get or set whether a track with an Instrument Rack device is currently showing its chains in Session View.

is_visible bool

Access: get

0 = track is hidden in a folded Group Track.

mute bool

Access: get set observe

[not in master track]

muted_via_solo bool

Access: get observe

1 = the track or chain is muted due to Solo being active on at least one other track.

name symbol

Access: get set observe

As shown in track header.

output_meter_left float

Access: get observe

Smoothed momentary peak value of left channel output meter, 0.0 to 1.0. For tracks with audio output only. This value corresponds to the meters shown in Live. Please take into account that the left/right audio meters add a significant load to Live GUI resource usage.

output_meter_level float

Access: get observe

Hold peak value of output meters of audio and MIDI tracks, 0.0 to 1.0. For audio tracks, it is the maximum of the left and right channels. The hold time is 1 second.

output_meter_right float

Access: get observe

Smoothed momentary peak value of right channel output meter, 0.0 to 1.0. For tracks with audio output only. This value corresponds to the meters shown in Live.

performance_impact float

Access: get observe

Reports the performance impact of this track.

output_routing_channel dictionary

Access: get set observe

The currently selected target channel for the track's output routing. It's represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the track's *available_output_routing_channels*.
Not available on the master track.

output_routing_type dictionary

Access: get set observe

The currently selected target type for the track's output routing. It's represented as a *dictionary* with the following keys:
`display_name` [symbol]
`identifier` [symbol]
Can be set to all values found in the track's *available_output_routing_types*.
Not available on the master track.

playing_slot_index int

Access: get observe

First slot has index 0, -2 = Clip Stop slot fired in Session View, -1 = Arrangement recording with no Session clip playing. [not in return/master tracks]

solo bool

Access: get set observe

Remark: when setting this property, the exclusive Solo logic is bypassed, so you have to unsolo the other tracks yourself. [not in master track]

Functions

create_audio_clip

Parameters:
`file_path` [symbol]
`position` [float]
Given an absolute path to a valid audio file in a supported format, creates an audio clip that references the file at the specified position in the arrangement view. Prints an error if the track is not an audio track, if the track is frozen, or if the track is being recorded into. The position must be within the range [0., 1576800].

See the `ClipSlot.create_audio_clip` function if you need to create audio clips in session view instead.

create_midi_clip

Parameters:
`start_time` [float]
`length` [float]
Creates an empty MIDI clip and inserts it into the arrangement at the specified time. Throws an error when called on a non-MIDI track or a frozen track, when the specified time is outside the [0., 1576800.] range, or when the track is currently being recorded into.

See the `ClipSlot.create_clip` function if you need to create audio clips in session view instead.

create_take_lane

Creates a take lane for this track.

delete_clip

Parameter: `clip`
Delete the given clip.

delete_device

Parameter: `index`
Delete the device at the given index.

duplicate_clip_slot

Parameter: `index`

Works like 'Duplicate' in a clip's context menu.

duplicate_clip_to_arrangement

Parameters: `clip``destination_time` [float]

Duplicate the given clip to the Arrangement, placing it at the given *destination_time* in beats. The source clip can be a session or an arrangement clip.

insert_device

Parameters: `device_name` [symbol] `target_index` [int] (optional)

Attempts to insert the device specified by `device_name` at the given index in the track's device chain. If no index is provided, attempts to insert the device at the end of the chain. Throws an error if insertion is not possible.
`device_name` is the name as it appears in the UI of Live.
Not all indices are valid. As can be expected, indices outside of the range defined by the current length of the device chain are invalid, but there are other limitations: for example, a MIDI effect can't be inserted after an instrument. The rule of thumb is that if an index would be invalid when inserting using the mouse, it's invalid here.

At the moment, only native Live devices can be inserted. Max for Live devices and plug-in are not supported.

Available since Live 12.3.

jump_in_running_session_clip

Parameter: `beats`

`beats` [float] is the amount to jump relatively to the current clip position.
Modify playback position in running Session clip, if any.

stop_all_clips

Stops all playing and fired clips in this track.

TuningSystem

Canonical Path 194
Properties 194
name 194
pseudo_octave_in_cents 195
lowest_note 195
highest_note 195
reference_pitch 195
note_tunings 195

This class represents a tuning system in Live.

Canonical Path

```
live_set tuning_system
```

Properties

name symbol

Access: get set observe

The name of the currently active tuning system.

pseudo_octave_in_cents float

Access: get

The pseudo octave in cents of the currently active tuning system.

lowest_note dictionary

Access: get set observe

The note index within the pseudo octave and octave of the lowest note.

highest_note dictionary

Access: get set observe

The note index within the pseudo octave and octave of the highest note.

reference_pitch dictionary

Access: get set observe

The reference pitch of the current tuning system.

note_tunings dictionary

Access: get set observe

The relative note tunings of the Tuning System in cents. Provided as a single-element dictionary holding an array.

WavetableDevice

Properties 196
filter_routing 197
mono_poly 197
oscillator_1_effect_mode 197
oscillator_2_effect_mode 197
oscillator_1_wavetable_category 197
oscillator_2_wavetable_category 198
oscillator_1_wavetable_index 198
oscillator_2_wavetable_index 198
oscillator_1_wavetables 198
oscillator_2_wavetables 198
oscillator_wavetable_categories 199
poly_voices 199
unison_mode 199
unison_voice_count 199
visible_modulation_target_names 200
Functions 200
add_parameter_to_modulation_matrix 200
get_modulation_target_parameter_name 200
get_modulation_value 200
is_parameter_modulatable 200
set_modulation_value 201

This class represents a Wavetable instrument.

A WavetableDevice shares all of the children, functions and properties that a Device has. Listed below are members unique to it.

Properties

filter_routing int

Access: get set observe

Access to the current filter routing. 0 = Serial, 1 = Parallel, 2 = Split.

mono_poly int

Access: get set observe

Access to Wavetable's Poly/Mono switch. 0 = Mono, 1 = Poly.

oscillator_1_effect_mode int

Access: get set observe

Access to oscillator 1's effect mode. 0 = None, 1 = Fm, 2 = Classic, 3 = Modern.

oscillator_2_effect_mode int

Access: get set observe

Access to oscillator 2's effect mode.

oscillator_1_wavetable_category

Access: get set observe

Access to oscillator 1's wavetable category selector.

oscillator_2_wavetable_category

Access: get set observe

Access to oscillator 2's wavetable category selector.

oscillator_1_wavetable_index

Access: get set observe

Access to oscillator 1's wavetable index selector.

oscillator_2_wavetable_index

Access: get set observe

Access to oscillator 2's wavetable index selector.

oscillator_1_wavetables StringVector

Access: get observe

List of names of the wavetables currently available for oscillator 1. Depends on the current wavetable category selection (see *oscillator_1_wavetable_category*).

oscillator_2_wavetables StringVector

Access: get observe

List of names of the wavetables currently available for oscillator 2. Depends on the current wavetable category selection (see *oscillator_2_wavetable_category*).

oscillator_wavetable_categories StringVector

Access: get

List of the names of the available wavetable categories.

poly_voices int

Access: get set observe

The current number of polyphonic voices.

unison_mode int

Access: get set observe

Access to Wavetable's unison mode parameter.

0 = None
1 = Classic
2 = Shimmer
3 = Noise
4 = Phase Sync
5 = Position Spread
6 = Random Note

unison_voice_count int

Access: get set observe

Access to the number of unison voices.

visible_modulation_target_names StringVector

Access: get observe

List of the names of modulation targets currently visible in the modulation matrix.

Functions

add_parameter_to_modulation_matrix

Parameter: `parameter_to_add` [DeviceParameter]
Add an instrument parameter to the modulation matrix. Only works for parameters that can be modulated (see *is_parameter_modulatable*).

get_modulation_target_parameter_name

Parameter: `index` [int]
Return the modulation target parameter name at *index* in the modulation matrix as a [symbol].

get_modulation_value

Parameters: `modulation_target_index` [int] `modulation_source_index` [int]
Return the amount of the modulation of the parameter at `modulation_target_index` by the modulation source at `modulation_source_index` in Wavetable's modulation matrix.

is_parameter_modulatable

Parameter: `parameter` [DeviceParameter]
1 = `parameter` can be modulated. Call this before `add_parameter_to_modulation_matrix`.

set_modulation_value

Parameters: `modulation_target_index` [int] `modulation_source_index` [int]
Set the amount of the modulation of the parameter at `modulation_target_index` by the modulation source at `modulation_source_index` in Wavetable's modulation matrix.

Credits

Cycling '74
440 N Barranca Ave #4074
Covina, CA 91723 USA

Copyright 2026 Cycling '74. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Cycling '74. Every effort has been made to ensure that the information in this document is accurate. Cycling '74 assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Except as permitted by such license, no part of this publication may be reproduced, edited, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, recording or otherwise, without the prior written permission of Cycling '74.

Contact Support: https://cycling74.com/support/contact