LOM - The Live Object Model

CONTENT
Introduction
Object Model Overview
FOR MUSO
Max For Live

LOM - The Live Object Model

Objects which conserve the Live API described by their structure, properties and functions.

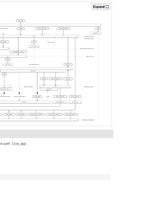
Introduction

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 chapter for definitions of the Basic Live API terms and a list of the Root objects used to access it.

This document refers to Ableton Live version 11.0

Object Model Overview

Click on the classes to navigate to their description.


Expand
Application
Application.View
Song
Control_Surface
Control_Surface.Component
Control_Surface.Component.Selection
Control_Surface.Component.Session_Navigation
Control_Surface.Component.Session_Ring
Control_Surface.Component.Transport
Control_Surface.Component.Undo_Redo
Control_Surface.Component.Viewport
Control_Surface.Component.Zoom
Control_Surface.Elements.Display_Data
Control_Surface.Elements.Encoder
Control_Surface.Elements.Button
Control_Surface.Elements.Slider
Control_Surface.Elements.Text_Display
Control_Surface.Elements.Touch_Strip
Control_Surface.Elements.Pad
Control_Surface.Elements.Color_Palette
Control_Surface.Elements.Color_Table
Control_Surface.Elements.Rgb_Color
Control_Surface.Elements.Midi_Map
Control_Surface.Elements.Raw_Midi_Map
Control_Surface.Elements.Scrollable
Control_Surface.Elements.Scroller
Control_Surface.Elements.Page
Control_Surface.Elements.Page_Manager
Control_Surface.Elements.Page_Component
Control_Surface.Elements.Page_Component_Manager
Control_Surface.Elements.Page_Stack
Control_Surface.Elements.Page_Stack_Component
Control_Surface.Elements.Page_Stack_Component_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager_Manager
Control_Surface.Elements.Page_Stack_Component_Manager
Control_Surface.Elements.Page_Stack_Component_Manager
Control_Surface.Elements.Page_Stack_Component_Manager
Control_Surface.Elements.Page_Stack_Component_Manager
Control_Surface.Elements.Page_Stack_Component_Manager

is_ableton_link_start_stop_sync_enabled

TYPE bool

ACCESS get, set, observe

Description

Enable/disable Ableton Link Start Stop Sync.

midi_recording_quantization

TYPE int

ACCESS get, set, observe

Description

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

nudge_down

TYPE bool

ACCESS get, set, observe

Description

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

nudge_up

TYPE bool

ACCESS get, set, observe

Description

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

tempo_follower_enabled

TYPE bool

ACCESS get, set, observe

Description

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

TYPE bool

ACCESS get, set, observe

Description

1 = MIDI Arrangement Overdub is enabled in the transport.

punch_in

TYPE bool

ACCESS get, set, observe

Description

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

punch_out

TYPE bool

ACCESS get, set, observe

Description

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

re_enable_automation_enabled

TYPE bool

ACCESS get, observe

Description

1 = the Re-Enable Automation button is on.

record_mode

TYPE bool

ACCESS get, set, observe

Description

1 = the Arrangement Record button is on.

root_note

TYPE int

ACCESS get, set, observe

Description

The root note of the song used for control surfaces. The root note can be a number between 0 and 11, where 0 = C and 11 = B.

scale_name

TYPE unicode

ACCESS get, set, observe

Description

The last used scale name used for control surfaces. The default scale names that can be saved with a Live Set are:

'Major'
'Minor'
'Dorian'
'Mixolydian'
'Lydian'
'Phrygian'
'Locrian'
'Diminished'
'Whole-half'
'Whole Tone'
'Minor Blues'
'Minor Pentatonic'
'Major Pentatonic'
'Harmonic Minor'
'Melodic Minor'
'Super Locrian'
'Bhairav'
'Hungarian Minor'
'Minor Gypsy'
'Hirojoshi'
'In-Sen'
'Iwato'
'Kumoi'
'Pelog'
'Spanish'

scale_intervals

TYPE list

ACCESS get, observe

Description

A list of integers representing the intervals in the current scale, expressed as the interval between the first scale degree and the scale degree at the list index.

select_on_launch

TYPE bool

ACCESS get

Description

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

session_automation_record

TYPE bool

ACCESS get, set, observe

Description

The state of the Automation Arm button.

session_record

TYPE bool

ACCESS get, set, observe

Description

The state of the Session Overdub button.

session_record_status

TYPE int

ACCESS get, observe

Description

Reflects the state of the Session Record button.

signature_denominator

TYPE int

ACCESS get, set, observe

Description

signature_numerator

TYPE int

ACCESS get, set, observe

Description

song_length

TYPE float

ACCESS get, observe

Description

A little more than `last_event_time` , in beats.

start_time

TYPE float

ACCESS get, set, observe

Description

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

swing_amount

TYPE float

ACCESS get, set, observe

Description

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

tempo

TYPE float

ACCESS get, set, observe

Description

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:frames* [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 [double] 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 [double] 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.

Song.View

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

Canonical path

live_set view

Children

detail_clip

TYPE Clip

ACCESS get, set, observe

Description

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

highlighted_clip_slot

TYPE ClipSlot

ACCESS get, set

Description

The slot highlighted in the Session View.

selected_chain

TYPE Chain

ACCESS get, set, observe

Description

The highlighted chain, or "id 0"

selected_parameter

TYPE DeviceParameter

ACCESS get, observe

Description

The selected parameter, or "id 0"

selected_scene

TYPE Scene

ACCESS get, set, observe

Description

selected_track

TYPE Track

ACCESS get, set, observe

Description

Properties

draw_mode

TYPE bool

ACCESS get, set, observe

Description

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

TYPE bool

ACCESS get, set, observe

Description

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.

GroovePool

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

TYPE list of Groove

ACCESS get, observe

Description

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

Track

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

clip_slots

TYPE list of ClipSlot

ACCESS get, observe

Description

arrangement_clips

TYPE list of Clip

ACCESS get, observe

Description

The list of this track's Arrangement View clip IDs

Available since Live 11.0.

devices

TYPE list of Device

ACCESS get, observe

Description

Includes mixer device.

group_track

TYPE Track

ACCESS get

Description

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

mixer_device

TYPE MixerDevice

ACCESS get

Description

view

TYPE Track.View

ACCESS get

Description

Properties

arm

TYPE bool

ACCESS get, set, observe

Description

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

available_input_routing_channels

TYPE dictionary

ACCESS get, observe

Description

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

TYPE dictionary

ACCESS get, observe

Description

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

TYPE dictionary

ACCESS get, observe

Description

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

TYPE dictionary

ACCESS get, observe

Description

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.

can_be_armed

TYPE bool

ACCESS get

Description

0 for return and master tracks.

can_be_frozen

TYPE bool

ACCESS get

Description

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

can_show_chains

TYPE bool

ACCESS get

Description

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

color

TYPE int

ACCESS get, set, observe

Description

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

TYPE long

ACCESS get, set, observe

Description

The color index of the track.

current_input_routing

TYPE symbol

ACCESS get, set, observe

Description

Like all other routings, reflects the string shown in the routing selector.

Deprecated since Live 9.7. Please use input_routing_type instead.

current_input_sub_routing

TYPE symbol

ACCESS get, set, observe

Description

Doesn't send any output if the selector has no entries.

Deprecated since Live 9.7. Please use input_routing_channel instead.

current_monitoring_state

TYPE int

ACCESS get, set, observe

Description

0 = In, 1 = Auto, 2 = Off [not in return/master tracks]

current_output_routing

TYPE symbol

ACCESS get, set, observe

Description

Deprecated since Live 9.7. Please use output_routing_type instead.

current_output_sub_routing

TYPE symbol

ACCESS get, set, observe

Description

Doesn't send any output if the selector has no entries.

Deprecated since Live 9.7. Please use output_routing_channel instead.

fired_slot_index

TYPE int

ACCESS get, observe

Description

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

TYPE int

ACCESS get, set

Description

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

TYPE bool

ACCESS get

Description

1 for audio tracks.

has_audio_output

TYPE bool

ACCESS get

Description

1 for audio tracks and MIDI tracks with instruments.

has_midi_input

TYPE bool

ACCESS get

Description

1 for MIDI tracks.

has_midi_output

TYPE bool

ACCESS get

Description

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

implicit_arm

TYPE bool

ACCESS get, set, observe

Description

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

input_meter_left

TYPE float

ACCESS get, observe

Description

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

TYPE float

ACCESS get, observe

Description

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

TYPE float

ACCESS get, observe

Description

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

TYPE dictionary

ACCESS get, set, observe

Description

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

TYPE dictionary

ACCESS get, set, observe

Description

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.

input_routings

TYPE list of symbols

ACCESS get

Description

Available input routings. [not in master track]

Deprecated since Live 9.7. Please use available_input_routing_types instead.

input_sub_routings

TYPE list of symbols

ACCESS get

Description

Available sub routings for the currently selected input routing. If no entries are available, nothing is output. [not in master track]

Deprecated since Live 9.7. Please use available_input_routing_channels instead.

is_foldable

TYPE bool

ACCESS get

Description

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

TYPE bool

ACCESS get, observe

Description

1 = the track is currently frozen.

is_grouped

TYPE bool

ACCESS get

Description

1 = the track is contained within a Group Track.

is_part_of_selection

TYPE bool

ACCESS get

Description

is_showing_chains

TYPE bool

ACCESS get, set, observe

Description

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

is_visible

TYPE bool

ACCESS get

Description

0 = track is hidden in a folded Group Track.

mute

TYPE bool

ACCESS get, set, observe

Description

[not in master track]

muted_via_solo

TYPE bool

ACCESS get, observe

Description

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

name

TYPE symbol

ACCESS get, set, observe

Description

As shown in track header.

output_meter_left

TYPE float

ACCESS get, observe

Description

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

TYPE float

ACCESS get, observe

Description

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

TYPE float

ACCESS get, observe

Description

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

TYPE float

ACCESS get, observe

Description

Reports the performance impact of this track.

output_routing_channel

TYPE dictionary

ACCESS get, set, observe

Description

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

TYPE dictionary

ACCESS get, set, observe

Description

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.

output_routings

TYPE list of symbols

ACCESS get

Description

Available output routings. [not in master track]

Deprecated since Live 9.7. Please use available_output_routing_types instead.

output_sub_routings

TYPE list of symbols

ACCESS get

Description

Available sub routings for the currently selected output routing. If no entries are available, nothing is output. [not in master track]

Deprecated since Live 9.7. Please use available_output_routing_channels instead.

playing_slot_index

TYPE int

ACCESS get, observe

Description

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]

time

TYPE `int`

ACCESS get, set, observe

Description

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

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

Parameter: clip
destination_time (double)
Duplicate the given clip to the Arrangement, placing it at the given destination_time in beats.

jump_in_running_session_clip

Parameter: beats
Jumps (relative) in the amount to jump relatively to the current clip position.
Modifies playback position in running Session clip, if any.

stop_all_clips

Stops all playing and fired clips in this track.

TrackView

Representing the view aspects of a track.

Canonical path

live_set tracks N view

Children

selected_device

TYPE `Device`

ACCESS get, observe

Description

The selected device or the first selected device (in case of multigroup selection).

Properties

device_insert_mode

TYPE `int`

ACCESS get, set, observe

Description

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

TYPE `bool`

ACCESS get, set, observe

Description

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

Functions

select_instrument

Returns bool (1 = there are no devices to select).
Selects track's instrument or first device, makes it visible and focuses on it.

ClipSlot

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

TYPE `Clip`

ACCESS get

Description

(if 0 if slot is empty)

Properties

color

TYPE `long`

ACCESS get, observe

Description

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

color_index

TYPE `long`

ACCESS get, observe

Description

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

controls_other_clips

TYPE `bool`

ACCESS get, observe

Description

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

TYPE `bool`

ACCESS get, observe

Description

1 = a clip exists in this clip slot.

has_stop_button

TYPE `bool`

ACCESS get, set, observe

Description

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

is_group_slot

TYPE `bool`

ACCESS get

Description

1 = this clip slot is a Group Track slot.

is_playing

TYPE `bool`

ACCESS get

Description

1 = playing, status != 0, otherwise 0.

is_recording

TYPE `bool`

ACCESS get

Description

1 = playing, status == 2, otherwise 0.

is_triggered

TYPE `bool`

ACCESS get, observe

Description

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

playing_status

TYPE `int`

ACCESS get, observe

Description

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

TYPE `bool`

ACCESS get

Description

1 = clip slot will record on start.

Functions

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)
Starts playing the contained clip, if any. Starts recording if slot is empty and track is armed. Starts recording of armed and empty slots in 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)
T = Live simulates pressing of Clip Launch button until the state is set to F 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.

Clip

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 path

live_set tracks N clip_slots M clip

Canonical path

live_set tracks N arrangement_clips M

Children

view

TYPE `ClipView`

ACCESS get

Description

Properties

available_warp_modes

TYPE `list`

ACCESS get

Description

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

color

TYPE `int`

ACCESS get, set, observe

Description

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

TYPE `int`

ACCESS get, set, observe

Description

The clip's color index.

end_marker

TYPE `double`

ACCESS get, set, observe

Description

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

end_time

TYPE `double`

ACCESS get, observe

Description

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

TYPE `double`

ACCESS get, set, observe

Description

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

gain_display_string

TYPE `symbol`

ACCESS get

Description

Get the gain display value of the clip as a string (e.g. "2.3 dB"). Can only be called on audio clips.

file_path

TYPE `symbol`

ACCESS get

Description

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

groove

TYPE `Groove`

ACCESS get, set, observe

Description

GetSetsHaveare access to the groove associated with this clip.

Available since Live 11.0.

has_envelopes

TYPE `bool`

ACCESS get, observe

Description

GetsHaveor whether the clip has any automation.

has_groove

TYPE `bool`

ACCESS get

Description

Returns true if a groove is associated with this clip.

Available since Live 11.0.

is_arrangement_clip

TYPE `bool`

ACCESS get

Description

1 = The clip is an Arrangement clip.
Applies either an Arrangement or a Session clip.

is_audio_clip

TYPE `bool`

ACCESS get

Description

0 = MIDI clip, 1 = audio clip

is_midi_clip

TYPE `bool`

ACCESS get

Description

The opposite of is_audio_clip.

is_overdubbing

TYPE `bool`

ACCESS get, observe

Description

1 = clip is overdubbing.

is_playing

TYPE `bool`

ACCESS get, set

Description

1 = clip is playing or retrigging.

is_recording

TYPE `bool`

ACCESS get, observe

Description

1 = clip is recording.

is_triggered

TYPE `bool`

ACCESS get

Description

1 = Clip Launch button is blinking.

launch_mode

TYPE `int`

ACCESS get, set, observe

Description

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

TYPE `int`

ACCESS get, set, observe

Description

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

TYPE `bool`

ACCESS get, set, observe

Description

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

Available since Live 11.0.

length

TYPE `double`

ACCESS get

Description

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

TYPE `double`

ACCESS get, set, observe

Description

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

loop_jump

TYPE `long`

ACCESS observe

Description

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

loop_start

TYPE `double`

ACCESS get, set, observe

Description

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

TYPE `bool`

access get, set, observe
Description
[-1-] clip is looped. Unwrapped audio cannot be looped.

muted

type bool
access get, set, observe
Description
[-1-] = muted (i.e. the Clip Activator button of the clip is off).

name

type symbol
access get, set, observe
Description

notes

type long
access observe
Description
Observer sends long when the list of notes changes.
Available for MIDI clips only.

warp_markers

type dictlong
access get, observe
Description
The Clip's Warp Markers as a dict. Observing this property bangs when the warp_markers changes.

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

type int
access get, set, observe
Description
Pitch shift in semitones ("Transpose"). -48 ... 48.
Available for audio clips only.

pitch_fine

type float
access get, set, observe
Description
Extra pitch shift in cents ("Detune"). -50 ... 49.
Available for audio clips only.

playing_position

type float
access get, observe
Description
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 DAW time scale at the top of the clip slot.

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

type long
access observe
Description
Observer sends bang when playing/trigger status changes.

position

type float
access get, observe
Description
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

type bool
access get, set, observe
Description
[-1-] = an audio clip's RAM switch is enabled.

sample_length

type int
access get
Description
Length of the Clip's sample, in samples.

sample_rate

type float
access get
Description
Get the Clip's sample rate.

signature_denominator

type int
access get, set, observe
Description

signature_numerator

type int
access get, set, observe
Description

start_marker

type double
access get, set, observe
Description
The start marker of the clip in beats, independent of the loop state. Cannot be set behind the end marker.

start_time

type double
access get
Description
The start time of the clip, relative to the global song time. For Session View clips, this is the time the clip was started. For Arrangement View clips, this is the offset within the arrangement. The value is in beats.

velocity_amount

type float
access get, set, observe
Description
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

type int
access get, set, observe
Description
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

type bool
access get, set, observe
Description
[-1-] = Warp switch is on.
Available for audio clips only.

will_record_on_start

type bool
access get
Description
[-1-] = 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=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 (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
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=a full range of velocity values 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 (0-127 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 must 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 be within the range [0..L], where L is the sample's length. The sample_length Clip property helps with this.
- The sample time must be between the left and right adjacent markers' respective sample times (this is a logical constraint).
- Within these constraints, there are limitations on the resulting segments' BPM. The allowed BPM ranges (LO, HI):

apply_note_modifications

Parameter:
dictionary
Key: "notes" (list of note dictionaries) as returned from get_notes_extended.
Note specification dictionaries parsed 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 11.2. 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.
Example:
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 (float)(optional)
transposition_amount (optional) (int)
Duplicates all notes matching the given note IDs.
Optionally, the destination time and transposition of the copying notes in the clip. Existing notes can be queried with get_notes_extended.
The selection of notes will be duplicated in deselecting the rest 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 11.2.2

duplicate_region

Parameter:
region_start (float)(int)
region_length (float)(int)
destination_time (float)(int)
pitch (int)(optional)
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 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's 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=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=a full range of velocity values 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": [int], "pitch": [int], ...}. 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 parsh performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.2

get_notes

Parameter:
pitch: (int)(double/int)
time_pitch: (int)
time_span: (double/int)
pitch_span: (int)
Returns a list of notes that start in the given area. The output is similar to get_selected_notes. time_time and time_span are given in beats.

Deprecated since Live 11.0. Please use get_notes_extended instead.

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=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=a full range of velocity values 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 pitch, which are the ID values you would like to pass to the function.

If you are 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": [int], "pitch": [int], ...}. 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 parsh performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0.

get_notes_extended

Parameters:
time_pitch: [int]
pitch_span: [int]
time_span: [double]
time_span: [float]

time_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=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=a full range of velocity values 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 single dictionary instead. The dictionary must include all of the parameters names given above as its keys; the associated values are the parameter values you wish to pass to the function.

If you are 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": [int], "pitch": [int], ...}. 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 parsh performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0. Replaces get_notes.

get_selected_notes

Use this if you want to operate on the selected notes.
Output:
get_selected_notes: notes: count
get_selected_notes: note: pitch time duration velocity mute

get_selected_notes: done

count (int) is the number of note lines to follow.
note (int) is the note number, 0..127, 60=C3
time (double) is the note start time in beats of absolute clip time.
duration (double) is the note length in beats.
velocity (int) is the note velocity, 0..127
mute (bool) 1 = the note is deactivated

Deprecated since Live 11.0. Please use get_selected_notes_extended instead.

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=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=a full range of velocity values 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": [int], "pitch": [int], ...}. 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 parsh performance when processing large notes dictionaries.

For MIDI clips only.

Available since Live 11.0. Replaces get_notes.

move_playing_pos

Parameter: beats:
Moves the playing position jump distance in beats. Negative beats jump backwards.
Jump by jump amount, unquantized.
Unwarped audio clips, recording audio clips and recording non-warped MIDI clips cannot jump.

move_warp_marker

Parameters: beat_time (double)
beat_time_distance (double)
Moves the warp marker specified by beat_time the specified beat time distance.

quantize

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

remove_notes

Parameter:
from_time [double]
time_span [double]
from_pitch [int]
pitch_span [int]

Deletes all notes that start in the given area. from_time and time_span are given in beats.

Deprecated since Live 11.0. Please use remove_notes_extended instead. If a device uses the deprecated method in Live 11, a warning pop-up will notify users that the device needs to be updated.

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_time [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: time [float]
Removes the warp marker at the given beat time.

replace_selected_notes

To replace the notes selected in the clip, send a series of the following function calls to the clip.

```
replace_selected_notes
notes count
note pitch time duration velocity muted
done
```

The sequence is the same as in get_selected_notes . the parameters are the same too. The sequence must always be completed, since the clip waits for the respective calls even if intermediate notes are sent. If there's none, this means there are no notes only.

In Live, notes don't overlap, so if the new notes overlap with unselected notes, the unselected ones are made to fit in the same way as when you edit notes manually.

An example sequence of calls looks like this:

```
call: replace_selected_notes
call: notes 3 0 0 0
call: note 36 0.5 0.1 100 0
call: note 42 0.5 0.1 50 0
call: done
```

For MIDI clips only.

Deprecated since Live 11.0. Please use apply_note_modifications instead. If a device uses the deprecated method in Live 11, a warning pop-up will notify users that the device needs to be updated.

scrub

Parameter: beat_time [double]
Scrub the clip in time, specified in beats. This behaves exactly like scrubbing with the mouse: the scrub will respect Global Quantization, starting and stopping in time with the grid. 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 11.0.0

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.

set_notes

Parameter:
list: of notes (pitch, time, duration, velocity, is_muted)
Will apply the given notes to the clip, overwriting existing notes.
An example sequence of calls looks like this:

```
call: set_notes
call: notes 2
call: note 36 0.5 0.1 100 0
call: note 42 0.5 0.1 50 0
call: done
```

For MIDI clips only.

Deprecated since Live 11.0. Please use add_new_notes or apply_note_modifications instead.

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.

ClipView

Representing the view aspects of a Clip.

Canonical path

live_set tracks N clip_slots M clip view

Properties

grid_is_triplet

TYPE bool
ACCESS get, set
Description
Get/set whether the clip is displayed with a triplet grid.

grid_quantization

TYPE int
ACCESS get, set
Description
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.

Groove

This class represents a groove in Live.

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

Canonical path

live_set groove_pool grooves N

Canonical path

live_set tracks N clip_slots M clip groove

Children

base

TYPE int
ACCESS get, set
Description
Get/set the groove's base grid (index based notion).
0 = 1/4
1 = 1/8
2 = 1/8T
3 = 1/16
4 = 1/16T
5 = 1/32

name

TYPE symbol
ACCESS get, set, observe
Description
Get/set/observe the name of the groove.

quantization_amount

TYPE float
ACCESS get, set, observe
Description
Get/set/observe the groove's quantization amount.

random_amount

TYPE float
ACCESS get, set, observe
Description
Get/set/observe the groove's random amount.

timing_amount

TYPE float
ACCESS get, set, observe
Description
Get/set/observe the groove's timing amount.

velocity_amount

TYPE float
ACCESS get, set, observe
Description
Get/set/observe the groove's velocity amount.

Device

This class represents a MIDI or audio device in Live.

Canonical path

live_set tracks N devices M

Canonical path

live_set tracks N devices M chains L devices K

Canonical path

live_set tracks N devices M return_chains L devices K

Children

parameters

TYPE list of DeviceParameter
ACCESS get, observe
Description
Only automatable parameters are accessible. See DeviceParameter to learn how to modify them.

view

TYPE DeviceView
ACCESS get
Description

Properties

can_have_chains

TYPE bool
ACCESS get
Description
Either a single device
Or a device Rack

can_have_drum_pads

TYPE bool
ACCESS get
Description
Like Drum Racks

class_display_name

TYPE symbol
ACCESS get
Description
Get the original name of the device (e.g. Operator , Auto Filter).

class_name

TYPE symbol
ACCESS get
Description
Live device type such as MultiBand , Operator , Limiter , ReducingAndAddTheC, or PluginDevice .

is_active

TYPE bool
ACCESS get, observe
Description
0 = either the device itself or its enclosing Rack device is off.

name

TYPE symbol
ACCESS get, set, observe
Description
This is the string shown in the title bar of the device.

type

TYPE symbol
ACCESS get
Description
The type of the device. Possible types are: 0 = undefined, 1 = instrument, 2 = audio_effect, 3 = midi_effect.

Functions

store_chosen_bank

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

DeviceView

Representing the view aspects of a Device.

Canonical path

live_set tracks N devices M view

Canonical path

live_set tracks N devices M chains L devices K view

Canonical path

live_set tracks N devices M return_chains L devices K view

Properties

is_collapsed

TYPE bool
ACCESS get, set, observe
Description
1 = the device is shown collapsed in the device chain.

DeviceParameter

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

TYPE int
ACCESS get, observe
Description
Get the automation state of the parameter.
0 = no automation
1 = automation active
2 = automation overridden.

default_value

TYPE float
ACCESS get
Description
Get the default value for this parameter.
Only available for parameters that aren't quantized (see is_quantized).

is_enabled

TYPE bool
ACCESS get
Description
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 (on thinks that they should not be moved).

is_quantized

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

max

TYPE float
ACCESS get
Description
Largest allowed value.

min

TYPE float
ACCESS get
Description
Lowest allowed value.

name

TYPE symbol
ACCESS get
Description
The short parameter name as shown in the (closed) automation chooser.

original_name

TYPE symbol
ACCESS get
Description
The name of a Macro parameter before its assignment.

state

TYPE int
ACCESS get, observe
Description
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

TYPE float
ACCESS get, set, observe
Description
Linear to DLS value between min and max.

value_items

TYPE StringVector
ACCESS get
Description
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

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

__str__

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

RackDevice

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 RackDevices.

Children

chains

TYPE list of Chain
ACCESS get, observe
Description
The Rack's chains.

drum_pads

TYPE list of DrumPad
ACCESS get, observe

Description

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

return_chains

TYPE list of Chain

ACCESS get, observe

Description

The Rack's return chains.

visible_drum_pads

TYPE list of DrumPad

ACCESS get, observe

Description

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

Properties

can_drum_chains

TYPE bool

ACCESS get

Description

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

has_drum_pads

TYPE bool

ACCESS get, observe

Description

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

TYPE bool

ACCESS get, observe

Description

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

is_showing_chains

TYPE bool

ACCESS get, set, observe

Description

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

variation_count

TYPE int

ACCESS get, observe

Description

The number of currently stored macro variations.

Available since Live 11.0.

selected_variation_index

TYPE int

ACCESS get, set

Description

Get/set the currently selected variation.

Available since Live 11.0.

Functions

copy_pad

Parameters: (int)
dest_pad_index, [int]
dest_variation_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.

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.

RackDeviceView

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

Children

selected_drum_pad

TYPE DrumPad

ACCESS get, set, observe

Description

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

selected_chain

TYPE Chain

ACCESS get, set, observe

Description

Currently selected chain.

Properties

drum_pads_scroll_position

TYPE int

ACCESS get, set, observe

Description

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

is_showing_chain_devices

TYPE bool

ACCESS get, set, observe

Description

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

DrumPad

This class represents a Drum Rack pad in Live.

Canonical path

live_set tracks N devices M drum_pads 0_

Children

chains

TYPE Chain

ACCESS get, observe

Description

Properties

mute

TYPE bool

ACCESS get, set, observe

Description

1 = muted

name

TYPE symbol

ACCESS get, observe

Description

note

TYPE int

ACCESS get

Description

solo

TYPE bool

ACCESS get, set, observe

Description

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

Functions

delete_all_chains

Chain

This class represents a group device chain in Live.

Canonical path

live_set tracks N devices M chains 1_

Canonical path

live_set tracks N devices M chains 1_devices 0_

Canonical path

live_set tracks N devices M return_chains 1_devices 0_chains P _

Children

devices

TYPE Device

ACCESS get, observe

Description

mixer_device

TYPE ChainMixerDevice

ACCESS get

Description

Properties

color

TYPE int

ACCESS get, set, observe

Description

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

TYPE long

ACCESS get, set, observe

Description

The color index of the chain.

is_arm_colored

TYPE bool

ACCESS get, set, observe

Description

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

has_audio_input

TYPE bool

ACCESS get

Description

has_audio_output

TYPE bool

ACCESS get

Description

has_midi_input

TYPE bool

ACCESS get

Description

has_midi_output

TYPE bool

ACCESS get

Description

mute

TYPE bool

ACCESS get, set, observe

Description

1 = muted (Chain Activator off)

muted_via_solo

TYPE bool

ACCESS get, observe

Description

1 = muted due to another chain being soloed.

name

TYPE unicode

ACCESS get, set, observe

Description

solo

TYPE bool

ACCESS get, set, observe

Description

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

Functions

delete_device

Parameters: (index [int])
Delete the device at the given index.

DrumChain

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

out_note

TYPE int

ACCESS get, set, observe

Description

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

choke_group

TYPE int

ACCESS get, set, observe

Description

Get/set the chain's choke group.

ChainMixerDevice

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

Canonical path

live_set tracks N devices M chains 1_mixer_device

Canonical path

live_set tracks N devices M return_chains 1_mixer_device

Children

sends

TYPE list of DeviceParameter

ACCESS get, observe

Description

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

chain_activator

TYPE DeviceParameter

ACCESS get

Description

panning

TYPE DeviceParameter

ACCESS get

Description

[In Audio Effect Racks and Instrument Racks only]

volume

TYPE DeviceParameter

ACCESS get

Description

[In Audio Effect Racks and Instrument Racks only]

SimplerDevice

This class represents an instance of Simpler.
A Simpler device 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

TYPE Sample

ACCESS get, observe

Description

The sample currently loaded into Simpler.

Properties

can_warp_as

TYPE bool

ACCESS get, observe

Description

1 = warp_as is available.

can_warp_double

TYPE bool

ACCESS get, observe

Description

1 = warp_double is available.

can_warp_half

TYPE bool

ACCESS get, observe

Description

1 = warp_half is available.

multi_sample_mode

TYPE bool

ACCESS get, observe

Description

1 = Simpler is in multisample mode.

pad_slicing

TYPE bool

ACCESS get, set, observe

Description

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

playback_mode

TYPE int

ACCESS get, set, observe

Description

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

playing_position

TYPE float

ACCESS get, observe

Description

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

playing_position_enabled

TYPE bool

ACCESS get, observe

Description:

1-> Simpler is playing back the sample and showing the playing position.

retrigger

TYPE bool

ACCESS get, set, observe

Description:

1-> Retrigger is enabled in Simpler.

slicing_playback_mode

TYPE int

ACCESS get, set, observe

Description:

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

voices

TYPE int

ACCESS get, set, observe

Description:

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 markers.

reverse

Reverse the loaded sample.

warp_as

Parameters: (bar) [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.

SimplerDeviceView

Represents the view aspects of a SimplerDevice.
SimplerDeviceView inherits from DeviceView, meaning that it has all the properties that a DeviceView has. Listed below are the members unique to SimplerDeviceView.

Properties

selected_slice

TYPE int

ACCESS get, set, observe

Description:

The currently selected slice, identified by its slice time.

Sample

This class represents a sample file loaded into Simpler.

Canonical path

live_set tracks 0 devices 0 sample

Properties

beats_granulation_resolution

TYPE int

ACCESS get, set, observe

Description:

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

TYPE float

ACCESS get, set, observe

Description:

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

0-> fastest decay
500-> no fade

beats_transient_loop_mode

TYPE int

ACCESS get, set, observe

Description:

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

TYPE float

ACCESS get, set, observe

Description:

Get/set the Envelope parameter in Complex Pro Mode.

complex_pro_formants

TYPE float

ACCESS get, set, observe

Description:

Get/set the Formants parameter in Complex Pro Mode.

end_marker

TYPE int

ACCESS get, set, observe

Description:

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

file_path

TYPE unicode

ACCESS get, observe

Description:

Get the path of the sample file.

gain

TYPE float

ACCESS get, set, observe

Description:

Get/set the sample gain.

length

TYPE int

ACCESS get

Description:

Get the length of the sample file in sample frames.

sample_rate

TYPE int

ACCESS get

Description:

The sample rate of the loaded sample.

Available since Live 11.0.

slices

TYPE list of int

ACCESS get, observe

Description:

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

TYPE float

ACCESS get, set, observe

Description:

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

start_marker

TYPE int

ACCESS get, set, observe

Description:

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

texture_flux

TYPE float

ACCESS get, set, observe

Description:

Get/set the Flux parameter in Texture Mode.

texture_grain_size

TYPE float

ACCESS get, set, observe

Description:

Get/set the Grain Size parameter in Texture Mode.

tones_grain_size

TYPE float

ACCESS get, set, observe

Description:

Get/set the Grain Size parameter in Tones Mode.

warp_markers

TYPE dict/beat

ACCESS get, observe

Description:

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

TYPE int

ACCESS get, set, observe

Description:

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

warping

TYPE bool

ACCESS get, set, observe

Description:

1-> warping is enabled.

slicing_style

TYPE int

ACCESS get, set, observe

Description:

Get/set the Slicing Mode.
0-> Transient
1-> Beat
2-> Region
3-> Manual

slicing_beat_division

TYPE int

ACCESS get, set, observe

Description:

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

TYPE int

ACCESS get, set, observe

Description:

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

Functions

get_display_string

Returns: (list of number) 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.

WavetableDevice

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

TYPE int

ACCESS get, set, observe

Description:

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

mono_poly

TYPE int

ACCESS get, set, observe

Description:

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

oscillator_1_effect_mode

TYPE int

ACCESS get, set, observe

Description:

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

oscillator_2_effect_mode

TYPE int

ACCESS get, set, observe

Description:

Access to oscillator 2's effect mode.

oscillator_1_wavetable_category

TYPE

ACCESS get, set, observe

Description:

Access to oscillator 1's wavetable category selector.

oscillator_2_wavetable_category

TYPE

ACCESS get, set, observe

Description:

Access to oscillator 2's wavetable category selector.

oscillator_1_wavetable_index

TYPE

ACCESS get, set, observe

Description:

Access to oscillator 1's wavetable index selector.

oscillator_2_wavetable_index

TYPE

ACCESS get, set, observe

Description:

Access to oscillator 2's wavetable index selector.

oscillator_1_wavetables

TYPE StringVector

ACCESS get, observe

Description:

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

TYPE StringVector

ACCESS get, observe

Description:

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

TYPE StringVector

ACCESS get

Description:

List of the names of the available wavetable categories.

poly_voices

TYPE int

ACCESS get, set, observe

Description:

The current number of polyphonic voices.

unison_mode

TYPE int

ACCESS get, set, observe

Description:

Access to Wavetable's unison mode parameter.

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

unison_voice_count

TYPE int

ACCESS get, set, observe

Description:

Access to the number of unison voices.

visible_modulation_target_names

TYPE StringVector

ACCESS get, observe

Description:

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

Functions

add_parameter_to_modulation_matrix

Parameters: (parameter) [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

Parameters: (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. Throws if Wavetable's modulation matrix.

is_parameter_modulatable

Parameters: (parameter) [DeviceParameter]

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 Wavemakers modulation matrix.

CompressorDevice

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

TYPE dict
ACCESS get, observe
Description:
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

TYPE dict
ACCESS get, observe
Description:
The list of available source types for the compressor's input routing in the sidechain. It's represented as a dictionary with the following keys:
available_input_routing_types: [list]
The list contains dictionaries as described in input_routing_type.

input_routing_channel

TYPE dict
ACCESS get, set, observe
Description:
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

TYPE dict
ACCESS get, set, observe
Description:
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

PluginDevice

This class represents a plug-in device.

Properties

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.

presets

TYPE StringVector
ACCESS get, observe
Description:
Get the list of the plug-in's presets.

selected_preset_index

TYPE int
ACCESS get, set, observe
Description:
Get/set the index of the currently selected preset.

MaxDevice

This class represents a Max for Live device in Live.

Properties

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.

audio_inputs

TYPE list of DeviceIO
ACCESS get, observe
Description:
List of the audio inputs that the MaxDevice offers.

audio_outputs

TYPE list of DeviceIO
ACCESS get, observe
Description:
List of the audio outputs that the MaxDevice offers.

midi_inputs

TYPE list of DeviceIO
ACCESS get, observe
Description:
List of the midi inputs that the MaxDevice offers.

Available since Live 11.0.

midi_outputs

TYPE list of DeviceIO
ACCESS get, observe
Description:
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 int] 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.

MixerDevice

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

TYPE list of DeviceParameter
ACCESS get, observe
Description:
One send per return track.

can_volume

TYPE DeviceParameter
ACCESS get
Description:
[in master track only]

crossfader

TYPE DeviceParameter
ACCESS get
Description:
[in master track only]

left_split_stereo

TYPE DeviceParameter
ACCESS get
Description:
The Track's Left Split Stereo Pan Parameter.

panning

TYPE DeviceParameter
ACCESS get
Description:

right_split_stereo

TYPE DeviceParameter
ACCESS get
Description:
The Track's Right Split Stereo Pan Parameter.

song_tempo

TYPE DeviceParameter
ACCESS get
Description:
[in master track only]

track_activator

TYPE DeviceParameter
ACCESS get
Description:

volume

TYPE DeviceParameter
ACCESS get
Description:

Properties

crossfade_assign

TYPE int
ACCESS get, set, observe
Description:
0 = A, 1 = none, 2 = B (not in master track)

panning_mode

TYPE int
ACCESS get, set, observe
Description:
Access to the Track mixer's pan mode: 0 = Stereo, 1 = Split Stereo.

EqDevice

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

Properties

edit_mode

TYPE bool
ACCESS get, set, observe
Description:
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 2-Mono: 0 = L, 1 = R
In MS mode: 0 = M, 1 = S
In Stereo mode: 0 = A, 1 = B (inactive)

global_mode

TYPE int
ACCESS get, set, observe
Description:
Access to EQ Eight's global mode. The modes are encoded as follows:

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

oversample

TYPE bool
ACCESS get, set, observe
Description:
Access to EQ Eight's Oversampling parameter. 0 = Off, 1 = On.

EqDeviceView

Represents the view aspects of an EqDevice.
An EqDeviceView has all the children, properties and functions of a DeviceView. Listed below are members unique to it.

Properties

selected_band

TYPE int
ACCESS get, set, observe
Description:
The index of the currently selected filter band.

HybridReverbDevice

This class represents an instance of a HybridReverb 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

TYPE float
ACCESS get, set, observe
Description:
The attack time of the amplitude envelope for the impulse response, in seconds.

ir_category_index

TYPE int
ACCESS get, set, observe
Description:
The index of the selected impulse response category.

ir_category_list

TYPE StringVector
ACCESS get
Description:
The list of impulse response categories.

ir_decay_time

TYPE float
ACCESS get, set, observe
Description:
The decay time of the amplitude envelope for the impulse response, in seconds.

ir_file_index

TYPE int
ACCESS get, set, observe
Description:
The index of the selected impulse response files from the current category.

ir_file_list

TYPE StringVector
ACCESS get, observe
Description:
The list of impulse response files from the selected category.

ir_size_factor

TYPE float
ACCESS get, set, observe
Description:
The relative size of the impulse response, 0.0 to 1.0.

ir_time_shaping_on

TYPE bool
ACCESS get, set, observe
Description:
Enables transforming the current selected impulse response with an amplitude envelope and size parameter.
1 = enabled.

SpectralResonatorDevice

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

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Freq control's mode.
0 = Hertz, 1 = MIDI note values.

midi_gate

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the MIDI gate switch's state.
0 = Off, 1 = On.

mod_mode

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Modulation Mode.
0 = None, 1 = Chorus, 2 = Wobble, 3 = Granular.

mono_poly

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Mono/Poly switch's state.
0 = Mono, 1 = Poly.

pitch_mode

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Pitch Mode.
0 = Internal, 1 = MIDI.

pitch_bend_range

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Pitch Bend Range.

polyphony

TYPE int
ACCESS get, set, observe
Description:
Get, set and observe the Polyphony.
0 = 1, 1 = 2, 2 = 4, 3 = 16 voices.

DeviceIO

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

Properties

available_routing_channels

TYPE dictionary
ACCESS get, observe
Description:
The available channels for this input/output box. 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

TYPE dictionary
ACCESS get, observe
Description:
The available types for this input/output box. 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

TYPE bool
ACCESS get, set
Description:
1 = the default routing channel for External routing types is none.

Available since Live 11.0.

routing_channel

TYPE dictionary
ACCESS get, set, observe
Description:
The current routing channel for this input/output box. 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

TYPE dictionary

ACCESS get, set, observe

Description

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*.

Scene

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

Canonical path

live_set scenes N

Children

clip_slots

TYPE list of ClipSlot

ACCESS get, observe

Description

Properties

color

TYPE int

ACCESS get, set, observe

Description

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

TYPE long

ACCESS get, set, observe

Description

The color index of the scene.

is_empty

TYPE bool

ACCESS get

Description

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

is_triggered

TYPE bool

ACCESS get, observe

Description

1 = scene is blinking.

name

TYPE symbol

ACCESS get, set, observe

Description

May contain BPM and signature commands (like `"128 BPM 3/4"`).

tempo

TYPE float

ACCESS get, set

Description

Tempo in BPM as found by Live in the name.

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.

CuePoint

Represents a locator in the Arrangement View.

Canonical path

live_set cue_points N

Properties

name

TYPE symbol

ACCESS get, observe

Description

time

TYPE float

ACCESS get, observe

Description

Arrangement position of the marker in beats.

Functions

jump

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

ControlSurface

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

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 from the control surface to Max for Live.

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.

this_device

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

See Also

Name	Description
Max For Live	Max For Live

Check out projects made by other Max users

OTHER RESOURCES

Cycling '74 Website

DOCUMENTATION

Max 8 Documentation
Max 7 Documentation
Max 6 Documentation
Max 5 Documentation

Terms and Conditions | © Cycling '74