Interface Specification

This document contains descriptions for all front-facing interface functions. Note that functions used solely by the back-end aren’t included.

# Exceptions

Exceptions that can be thrown by the backend.

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Usage** |
| ProjectReadError | Thrown when a project is in an invalid format |  |
|  |  |  |

# Types

Data types used by the back end. Use structs not classes to pass by copy.

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Usage** |
| MidiLength | Length of a MIDI object (in ticks) | Lengths of notes, phrases |
| MidiPos | Position of a MIDI object (in ticks) relative to the parent body | Positions of MIDI notes relative to the start of a phrase, positions of phrases relative to the start of the project |
| MidiCoord | Position in a phrase or project (x (MidiPos) and y (int) coordinates) | Selection points, trim points |
| MidiBox | Contains two non-overlapping MidiCoord structs | Trimming of phrases, bounding boxes of clips |

# Objects

Settings

Contains the program’s settings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| saveProject | Saves a project (if it has been changed) | (none) | (none) | IOError: error saving project file |

Projects

Methods related to multiple project management, contained inside the static class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| addProject | Create a new project | (none) | (int) projID: identifier that can be used to refer to this project | (none) |
| Open an existing project | (str) projDir: directory of the project to open | IOError: the directory doesn’t exist  ProjectReadError: the project folder doesn’t contain a valid file |
| Open an existing compressed project | (str) projFile: file location of the file to load | IOError: error reading file (file doesn’t exist or access denied)  ProjectReadError: the project file (when decompressed) doesn’t contain a valid file |
| closeProject | Close an open project (without saving) | (int) projID: project’s identifier | (none) | (none) |
| saveAllProjects | Saves every project that requires it | (none) | (none) | IOError: error saving project file(s) |

Project

Methods related to a single project, contained inside the class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| saveProject | Saves a project (if it has been changed) | (none) | (none) | IOError: error saving project file |
| exportProject | Compiles a project to a MIDI file | (str) fileName | (none) | IOError: error saving MIDI file |

Details

Properties related to a project’s details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| Name | **Name of the project (str)** |  |  |  |
| Author | **Author of the project (str)** |  |  |  |
| Description | **Description of the project (str)** |  |  |  |
| Copyright | **Copyright information (str)** |  |  |  |
| getWorkingTime | **Returns the total time spent working on the project.** | **(none)** | **(**TimeDiff**) working time** |  |

History

Methods related to a project’s history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| undo |  |  |  | InvalidOperationException: nothing to undo |
| redo |  |  |  | InvalidOperationException: nothing to redo |
| UndoPos | **Get or set position in undo history** |  |  | ArgumentOutOfRangeException: position in undo history doesn’t exist |

Phrases

Methods relating to phrase management in a project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| newPhrase | **Create a new phrase** | **(none)** | **(int)** phraseID |  |
| importPhrase | **Import MIDI file as phrase** | **(str)** fileName | **(int)** phraseID | IOError: error reading MIDI file (file doesn’t exist or access denied) |
| removePhrase | **Remove a phrase** | **(int)** phraseID | **(none)** |  |
| duplicatePhrase | **Make a copy of a phrase** | **(int)** phraseID | **(int)** newPhraseID |  |

Phrase

Methods and properties relating to a single phrase object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| Name | **Get or set the name of the phrase** | (str) | **(str)** | ArgumentError: newName is empty |
| previewImg | **Returns the file address of the preview image (file location based on hash of file)** | (none) | **(str) file address** |  |
| getLength | Returns the length of a phrase | (none) | (MidiLength) length |  |
| getHeight | Returns the number of tracks in a phrase | (none) | (int) number of tracks |  |
| File | Get or set the address of the MIDI file associated with the phrase | (str) file address | |  |
| Loaded (get, set) | Whether the phrase is loaded into memory for MIDI playback | (bool) | |  |
| Muted (get, set) | Whether the phrase will be included in playback | (bool) | |  |
| Colour (get, set) | Colour used when displaying phrase | (Colour) | |  |

Clip

Methods related to an instance of a phrase on the timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Description** | **Arguments** | **Returns** | **Exceptions** |
| Anchor | **Location of clip on the timeline (get, set)** | (MidiCoord) | **(**MidiCoord**)** |  |
| Trim | **Trimming of the phrase (from top left and bottom right of clip). Ensure that when updating the top left, the A**nchor **is also updated so that the instance doesn’t move around** | (MidiBox) | (MidiBox) | ArgumentError: attempt to trim past the end or before beginning of clip  ArgumentError: top left and bottom right are ordered incorrectly |
| getCoords | **Returns the bounding box of the phrase instance relative to the timeline.** | (none) | (MidiBox) |  |