Developer Reference

This page documents PyFLP's internals which consists of `pyflp._events`, `pyflp._descriptors` and `pyflp._models`.

> The content below assumes you have fairly good knowledge of the following:
>
> - OOP and descriptors, especially
> - Type annotations
> - Binary data types and streams

Events

Source code: pyflp/_events.py

Contains implementations for various types of event data and its container.

These types serve as the backbone for model creation and simplify marshalling and unmarshalling.

If you have read Part I, you know how events use a TLV encoding scheme.

Type

The `type` represents the event ID. A custom enum class (and a metaclass) supporting unknown IDs and member check using Python's `... in ...` syntax is used.

class pyflp._events.**_EventEnumMeta**(cls, bases, classdict, *, boundary=None, _simple=False, **kwds) [source]

class pyflp._events.**EventEnum** [source]

IDs used by events.

Event values are stored as a tuple of event ID and its designated type. The types are used to serialise/deserialise events by the parser.

All event names prefixed with an underscore (_) are deprecated w.r.t to the latest version of FL Studio, *to the best of my knowledge*.

Length

The `length` is a field prefixed for IDs in the range of 192-255. It is the size of `value` and is encoded as a VLQ128 (variable length quantity base-128).

Value

Below are the list of classes PyFLP has, grouped w.r.t the ID range.

0-63

64-127

128-191

192-255

EventTree

```
class pyflp._events.EventTree(parent: EventTree | None = None, init: Iterable[IndexedEvent] |
    None = None)                                                                    [source]
```

Provides mutable “views” which propagate changes back to parents.

This tree is analogous to the hierarchy used by models.

parent

Immediate ancestor / parent. Defaults to self.

root

Parent of all parent trees.

children

List of children.

`__iter__() → Iterator[EventBase[Any]]` [source]

`__len__() → int` [source]

`append(event: EventBase[Any]) → None` [source]

Appends an event at its corresponding key’s list’s end.

`count(id: EventEnum) → int` [source]

Returns the count of the events with `id`.

`divide(*args: ~typing.~P, **kwds: ~typing.~P)` [source]

`first(id: EventEnum) → EventBase[Any]` [source]

Returns the first event with `id`.

RAISES:

KeyError – An event with `id` isn’t found.

`get(*ids: EventEnum) → Iterator[EventBase[Any]]` [source]

Yields events whose ID is one of `ids`.

`group(*args: ~typing.~P, **kwds: ~typing.~P)` [source]

`property ids: frozenset[pyflp._events.EventEnum]`

`property indexes: frozenset[int]`

Returns root indexes for all events in `self`.

`insert(pos: int, e: EventBase[Any]) → None` [source]

Inserts `ev` at `pos` in this and all parent trees.

`pop(id: EventEnum, pos: int = 0) → EventBase[Any]`

Pops the event with `id` at `pos` in `self` and all parents.

`remove(id: EventEnum, pos: int = 0) → None` [source]

Removes the event with `id` at `pos` in `self` and all parents.

`separate(*args: ~typing.~P, **kwds: ~typing.~P)` [source]

`subtree(select: Callable[[EventBase[Any]], bool | None]) → EventTree` [source]

Returns a mutable view containing events for which `select` was True.

> **Caution**
>
> Always use this function to create a mutable view. Maintaining chilren and passing parent to a child are best done here.

`subtrees(*args: ~typing.~P, **kwds: ~typing.~P)` [source]

Models

Source code: pyflp/_models.py

Contains the ABCs used by model classes and some shared classes.

Implementing a model

A look at the **source code** will definitely help, although these are a few points that must be kept in mind when Implementing a model:

1. Does the model mimic the hierarchy exposed by FL Studio's GUI?

> **Tip**
>
> Browse through the hierarchies of `pyflp.channel.Channel` subclasses to get a very good idea of this.

2. Are `__dunder__` methods provided by Python used whenever possible?
3. Is either `ModelReprMixin` subclassed or `__repr__` implemented?

Descriptors

Source code: pyflp/_descriptors.py

Contains the descriptor and adaptor classes used by models and events.

Adapters

Source code: pyflp/_adapters.py