Index

Functions

Classes

Module **connection**

▶ EXPAND SOURCE CODE

Functions

`def create_dispatcher(settings_queues, channels)`

Creates a dispatcher to map incoming OSC messages into functions.

Args:

`settings_queues` : Multiprocessing queues in which to put incoming settings.
`channels` : Max number of channels currently being processed.

Returns:

A dispatcher for an OSC server.

▶ EXPAND SOURCE CODE

`def default_handler(address, *args)`

Handles OSC messages with unrecognized addresses.

Args: `address` : OSC address of the received message. `*args` : Arguments of the OSC message.

▶ EXPAND SOURCE CODE

`def handler_ch_settings(address, fixed_args, *osc_args)`

Handles an incoming OSC message to set settings.

Args: `address` : OSC address of the received message. `fixed_args` : Function arguments passed down from the calling higher function. `*osc_args` : Arguments of the OSC message.

▶ EXPAND SOURCE CODE

`def handler_start(address, *args)`

Starts execution of the application when a start message is received

Args: `address` : OSC address of the received message. `*args` : Arguments of the OSC message.

▶ EXPAND SOURCE CODE

`def handler_stop(address, *args)`

Stops execution of the application when a stop message is received

Args: `address` : OSC address of the received message. `*args` : Arguments of the OSC message.

▶ EXPAND SOURCE CODE

Classes

`class ConnectionHandler (address: str, port: int)`

Abstract class to handle the connection between the python script and the external world. Can be inherited to implement custom methods of sending messages.

Constructor for the ConnectionHandler class.

Args:

`address` : Net address of the receiver as a string.

`port` : Net port of the receiver as an int.

Class Attributes:

`_address` : Net address of the receiver as a string.

`_port` : Net port of the receiver as an int.

`_lock` : Mutex lock used for synchronization purposes.

▶ EXPAND SOURCE CODE

Ancestors

abc.ABC

Subclasses

OSCConnectionHandler

Static methods

`def get_instance(address: str, port: int) -> ConnectionHandler`

Abstract method to get the Singleton instance of the class.

Args:

`address` : Net address to assign to the ConnectionHandler Singleton instance.

`address` : Net port to assign to the ConnectionHandler Singleton instance.

Returns:

The Singleton instance of the class.

▶ EXPAND SOURCE CODE

Methods

`def send_message(self, message: Message)`

Abstract method used to send a message.

Args:

`message` : Message to send over the network.

▶ EXPAND SOURCE CODE

`class HFAudioMessage (data, channel: int)`

Message containing High-level Features.

Constructor for the HFAudioMessage class.

Args:

`data` : Data to send.

`channel` : Index of the track or input channel.

▶ EXPAND SOURCE CODE

Ancestors

Message, abc.ABC

Methods

`def to_osc(self) -> pythonosc.osc_message.OscMessage`

Converts message into its `OSC` representation with its own `OSC` address.

Returns:

The `OSC` representation of the message.

▶ EXPAND SOURCE CODE

`class LFAudioMessage (data, channel: int, instrument: Instruments)`

Message containing Low-level Features.

Constructor for the LFAudioMessage class.

Args:

`data` : Data to send.

`channel` : Index of the track or input channel.

`instrument` : Instrument of the channel.

▶ EXPAND SOURCE CODE

Ancestors

Message, abc.ABC

Methods

`def to_osc(self) -> pythonosc.osc_message.OscMessage`

Converts message into its OSC representation with its own OSC address. Appends the instrument type as a string argument and all the Low-level features as floats.

Returns:

The `OSC` representation of the message.

▶ EXPAND SOURCE CODE

`class Message (data, channel: int)`

Abstract Class representing the message with output data to be sent to the visualizer. Can be inherited to implement custom message types with custom parameters.

Constructor for the Message class.

Args:

`data` : Data to send.

`channel` : Index of the track or input channel.

Class Attributes:

`_data` : Data to send.

`channel` : Index of the track or input channel.

`address` : `OSC` address of the message.

▶ EXPAND SOURCE CODE

Ancestors

abc.ABC

Subclasses

HFAudioMessage, LFAudioMessage

Methods

`def to_osc(self) -> pythonosc.osc_message.OscMessage`

Abstract method to convert a message into its `OSC` representation.

Returns:

The `OSC` representation of the message.

▶ EXPAND SOURCE CODE

`class OSCConnectionHandler (address: str, port: int)`

Singleton class that handles the communication between the python script and the external world via `OSC` messages.

Singleton constructor. Starts the `OSC` communication channel.

Args:

`address` : Net address to assign to the ConnectionHandler Singleton instance.

`address` : Net port to assign to the ConnectionHandler Singleton instance.

▶ EXPAND SOURCE CODE

Ancestors

ConnectionHandler, abc.ABC

Static methods

`def get_instance(address: str, port: int) -> OSCConnectionHandler`

Returns the currently running Singleton Instance of the class.

Args:

`address` : Net address to assign to the ConnectionHandler Singleton instance.

`address` : Net port to assign to the ConnectionHandler Singleton instance.

Returns:

The Singleton instance of the class.

▶ EXPAND SOURCE CODE

Methods

`def send_message(self, message: Message)`

Sends a message over the network as an `OSC` message.

Args:

`message` : Message to be sent.

▶ EXPAND SOURCE CODE